Trabajo Fin de Máster
Máster Universitario en Ingeniería de Telecomunicación

Estudio e implementación de un proceso de Integración Continua

Autor: Lourdes Liró Salinas
Tutor: Germán Madinabeitia Luque

Dpto. de Ingeniería de Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2022

Trabajo Fin de Máster
Máster Universitario en Ingeniería de Telecomunicación

Estudio e implementación de un proceso de Integración Continua

Autor:
Lourdes Liró Salinas

Tutor:
Germán Madinabeitia Luque
Profesor Colaborador

Dpto. de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2022

Trabajo Fin de Máster: Estudio e implementación de un proceso de Integración Continua

Autor: Lourdes Liró Salinas

Tutor: Germán Madinabeitia Luque

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2022

El Secretario del Tribunal

A mis padres.

Agradecimientos

Gracias a mi familia por soportarme y a mis amigos por aguantarme.

Lourdes Liró Salinas

Sevilla, 2022

Resumen

En la actualidad la tendencia al desarrollo de proyectos software siguiendo la metodología trabajo Agile implica un ritmo de trabajo basado en la respuesta al cambio, lo que requiere que las personas responsables del desarrollo de aplicaciones, aseguramiento de calidad del software y gestión de sistemas colaboren de forma continua entre ellos y con el cliente y tiendan a automatizar cada vez más la integración de los procesos involucrados en la puesta en producción de este software.

A partir de esta situación surgen conceptos como la Integración Continua (IC) o DevOps.

En este documento se estudian las herramientas que existen actualmente para la facilitación y automatización de los procesos que permiten optimizar la puesta en producción de una aplicación partiendo de la realización de un cambio en el código y asegurando la calidad de los cambios.

Abstract

Nowadays, software projects tend to follow Agile methodologies which implies a work pace based on changes response. This tendency requires the continuous collaboration between people in charge of development, software quality, systems administration and clients. Automatization of the different processes integration involved becomes a need that ease software releases.

Continuous Integration and DevOps are some of the concepts that arise from this approach.

This study makes a comparison between tools used in order to automatize processes that allow the optimization of a software release starting from a code change and ensuring software quality.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE IMÁGENES

1 INTRODUCCIÓN

"Imagination is more important than knowledge. Knowledge is limited. Imagination encircles the world."

- Albert Einstein-

Este apartado que introduce al concepto de Integración Continua se divide en varias secciones con el objetivo de distribuir la información de forma que permita al lector un acercamiento deductivo al tema tratado. Comienza describiendo las distintas fases y usos de la Integración Continua, para continuar con varios conceptos que están bastante relacionados con la misma: Despliegue Continuo, Agile y DevOps.

1.1 Integración Continua

La Integración Continua, a partir de ahora IC, es una práctica de desarrollo software que consiste en que el trabajo realizado por los desarrolladores sea integrado frecuentemente y verificado mediante una compilación automática y ciertas pruebas que permitan que se detecten posibles errores lo antes posible, reduciendo así el riesgo de un proyecto software. Una de las más conocidas definiciones de este concepto y las prácticas recomendadas para llevarlo a cabo se describen en el artículo de Martin Fowler, "Continuous Integration" [1]; así como en el libro "Continuous Integration" [2], que recoge las ideas ya plasmadas en el artículo y añade más información al respecto.

Las prácticas que se recomiendan en las fuentes citadas y que permiten que se entienda de forma más práctica el concepto de IC, son las siguientes:

1. Mantener un único repositorio de código fuente asegurando que todo el equipo lo conozca bien, incluyendo lo necesario para compilar el software en local, manteniendo el uso de ramas al mínimo y siendo la rama principal la más utilizada.
2. Automatizar la compilación para que sea posible realizarla mediante un script y un solo comando.
3. Crear pruebas para el software que se desarrolla y hacerlo si es posible antes de desarrollar el código, ya que esto también ayuda a diseñar el software.
4. Publicación de cambios por parte de todos los desarrolladores cada día en la rama principal lo que se traduce en una comunicación frecuente entre desarrolladores que permite conocer los cambios en menos tiempo.
5. Cada cambio publicado debe lanzar una compilación en la máquina de integración lo cual asegura que la compilación funciona. Permite evitar problemas debido a la diferencia de entornos utilizados para el desarrollo. Se puede realizar de forma manual en una máquina de integración o se puede utilizar un servidor de IC que monitorice el repositorio y lance una compilación cada vez que se suba código a la rama principal.

6. Se deben solucionar los errores de compilación inmediatamente y no se debe depurar la rama principal con errores, sino que se deben deshacer los cambios que han producido el error y depurar en un entorno de desarrollo.
7. La compilación se debe realizar rápido; concretamente, se recomienda un tiempo de compilación menor a los diez minutos, que permita que el desarrollador pueda realizar más subidas de código en menos tiempo.
8. Las pruebas se deben realizar en un entorno lo más parecido posible al de producción ya que el objetivo de las pruebas es encontrar problemas que el sistema pueda presentar en producción.
9. Debe ser fácil para todos obtener el último ejecutable generado, para lo que se debe dejar en un lugar conocido por todos.
10. Todos los componentes del equipo deben poder ver qué está pasando, es decir, puedan saber si hay una compilación en progreso y el estado de la última compilación.
11. Se debe automatizar el despliegue y la marcha atrás. Esto se puede hacer mediante scripts.

En resumen, llevar a cabo todas estas prácticas o algunas de ellas, permite reducir los riesgos del proyecto, ya que disminuye la incertidumbre: siempre sabes qué funciona y qué no. Evita que se realice un desarrollo y que se integren cada una de las partes al final sin que se sepa qué va a fallar.

Por otra parte, la IC consigue que se realice una más rápida identificación de errores, lo que permite que su resolución sea menos compleja.

Por último, hace posible que se realicen despliegues de forma más frecuente, lo cual se traduce en que el usuario final o cliente puede ver antes las nuevas funcionalidades y realizar los comentarios que considere, creándose así un ambiente de desarrollo software mucho más ágil y colaborativo. Esto último encaja con el concepto de desarrollo Agile, el cual se describe un poco más adelante en el apartado 1.1.2.2 de este documento.

1.1.1 Diferencia entre IC y DC

Cabe diferenciar la Integración Continua del concepto de Despliegue Continuo [3]. El segundo enfoque consiste en permitir que el equipo software desarrolle en ciclos cortos y sea capaz de liberar entregas software para su despliegue en cualquier momento; es decir, es la parte que vendría después de la Integración Continua.

Parece importante adoptar la Integración Continua para que el Despliegue Continuo pueda funcionar, pero también lo es al contrario, ya que, ¿para qué una IC software si luego no se es capaz de desplegar en entornos productivos frecuentemente?

Aquí es donde entra en juego la posibilidad de automatizar también el despliegue en distintos entornos de la aplicación integrada y probada [4]. Para ello hay que tener en cuenta las configuraciones de los distintos entornos (por ejemplo: desarrollo, preproducción y producción) y la gestión de las variables de entorno asociadas a cada uno de ellos. Por ejemplo: la cadena de conexión a base de datos no será la misma para un entorno de desarrollo que para un entorno de producción.

1.1.2 La relación de la IC con DevOps y Agile

1.1.2.1 DevOps

DevOps es un término derivado de la unión de las palabras “developer” y “operations” o lo que es lo mismo “desarrollador” y “operaciones”. Esta combinación pretende representar un movimiento que nace de la necesidad de reducir barreras entre los equipos de desarrollo y operaciones para así conseguir que el tiempo de puesta en producción del software disminuya. DevOps implementa las prácticas que comprenden la IC y el DC para asegurar que el software se entrega rápido y con calidad. La eliminación de estas barreras recae en la identificación de la figura del denominado Ingeniero DevOps. El Ingeniero DevOps realiza tareas propias de la

gestión de operaciones y del desarrollo, debe conocer ambas disciplinas y, en la mayoría de las ocasiones, hacerse responsable del proceso de IC y DC; así como de la gestión de las infraestructuras [5].

La filosofía DevOps se presentó en 2008, en la conferencia Agile en Toronto [6], lo cual pone de manifiesto una vez más la estrecha relación entre estos dos conceptos.

1.1.2.2 Agile

Agile [7] es una metodología de gestión y desarrollo software que propone un conjunto de principios en los que:

- Se valoran más los individuos e interacciones que los procesos y herramientas.
- Se valora más el software en funcionamiento que la documentación excesiva
- Se valora más la colaboración con el cliente que la negociación contractual.
- Se valora más la respuesta ante al cambio al seguimiento de un plan.

Uno de los padres del manifiesto Agile, y quién le puso nombre al mismo [8], es Martin Fowler, de cuyo artículo [1] se han extraído las buenas prácticas de la IC que se han descrito un poco más arriba. Es por tanto el responsable de que estos dos conceptos estén tan relacionados y escribe en su artículo: "…una de las partes más difíciles del desarrollo software es asegurarse de que se está haciendo el software correcto. Nos resulta muy difícil especificar qué queremos que se haga; para las personas es mucho más fácil ver algo que no está bien hecho y decir qué necesitan que se cambie. Los procesos de desarrollo Agile esperan y aprovechan este comportamiento humano", con esto quiere decir que el proceso de IC hace que se puedan mostrar los últimos resultados del desarrollo con mucha frecuencia y permite que se hagan efectivas las buenas prácticas de Agile.

2 ORGANIZACIÓN DE LA MEMORIA

"Imagination is more important than knowledge. Knowledge is limited. Imagination encircles the world."

- Albert Einstein-

Este documento refleja el estudio realizado de forma previa a la implantación de un entorno que permita la realización del proceso de Integración Continua para una aplicación web así como los detalles de su implementación y las conclusiones extraídas.

Para ello ha sido necesaria la búsqueda y lectura de documentación que refleja el contexto en el que se encuentra la investigación en este campo y los avances logrados hasta la actualidad. Esto ha permitido obtener una visión general del estado del arte en el que se encuentra, el cual proporciona información que servirá de base para el resto del estudio.

Tras ello, se realiza una investigación sobre las herramientas y el entorno idóneos para hacer posible la puesta en práctica de la implementación elegida. Todo el software y herramientas estudiadas se describen en profundidad en *Análisis y comparativa de herramientas* y *Herramientas y entorno seleccionado*.

Una vez establecido el entorno y las herramientas necesarias, se implementa la solución en un entorno de verificación y se realizan los ajustes y pruebas necesarias para su implementación en un entorno real del cual se realizará un uso continuo. El lector puede consultar la información sobre el entorno y las pruebas realizadas en *Implementación* y *Pruebas realizadas*.

Para poder llevar a cabo la solución propuesta es necesario desarrollar una aplicación sobre la que realizar las pruebas, por lo que se ha desarrollado una aplicación web Java que consiste en la gestión, creación y borrado de datos de estudiantes (nombre, apellidos, fecha de nacimiento y correo electrónico) a través de una interfaz web. Esta aplicación se describe en el subapartado de *Implementación*: *Aplicación Java*.

Por último, a partir de la observación de los resultados y los problemas encontrados, se extraen conclusiones sobre el funcionamiento de las herramientas y posibles optimizaciones. Las conclusiones extraídas de este estudio se pueden leer en *Conclusiones*.

3 ESTADO DEL ARTE

"The important thing is not to stop questioning. Curiosity has its own reason for existence."

- Albert Einstein -

Para el estudio del estado del arte se han utilizado varias referencias relacionadas con la Integración Continua, el concepto DevOps y propuestas de entornos de Integración Continua. Gracias a ello, se obtiene un contexto que permite avanzar en el campo deseado y partir de una base.

3.1 Análisis y comparativa de herramientas

Partiendo de los conceptos ya revisados y de las necesidades que se plantean, se describen las herramientas que en la actualidad existen en el mercado para poder llevar a cabo el proceso de Integración Continua software. También se describen algunas herramientas que podrían formar parte del proceso de Despliegue Continuo y algunas que ayudan a implementar la filosofía DevOps y a seguir la metodología Agile. Esto es porque hoy en día estos conceptos suelen ir de la mano como ya se sugiere anteriormente en este estudio.

En primer lugar se van a plantear varias herramientas de Control de Versiones. Estas facilitarán el trabajo colaborativo entre desarrolladores y será clave para el proceso de IC ya que permitirá que se haga un seguimiento del código y que se integre con menos dificultad.

También se necesita compilar cada vez que se genere nuevo código para poder desplegar lo último que se ha integrado de forma continua. Para ello se presentarán varias herramientas para realizar esta compilación y la generación del empaquetado.

La calidad de la que se habla en DevOps depende en gran parte de la realización de pruebas, por lo que se describen varias herramientas para la realización de pruebas unitarias y de código estático.

Para continuar se describirá una de las herramientas más importantes en este proceso: el servidor de IC. Aquí es donde más barreras se eliminan entre los conceptos de desarrollo, operaciones y calidad. Permitirá automatizar el proceso desde que se realiza una subida de código hasta que se despliega en el servidor. Esto además facilitará que se siga la metodología Agile, ya que el cliente podrá acceder en cualquier momento al entorno y ver los últimos cambios integrados.

Si hablamos de DevOps, será indispensable mencionar varias herramientas de gestión de configuración, ya que hacen posible definir las distintas máquinas que intervienen en el proceso de IC y DC. Permitirán la gestión de estas máquinas y de los despliegues.

Por último, para que todo el mundo pueda acceder a los ejecutables generados, se plantearán distintas

herramientas de gestión de repositorios. Esto también mejorará el tiempo de generación de binarios permitiendo una integración más rápida.

3.1.1 Sistemas de Control de Versiones (VCS)

La gestión de los cambios en el código y el control de las versiones que se desarrollan en un proyecto software pueden ser gestionados de forma manual, pero es recomendable utilizar alguna de las herramientas que ya existen. Estas herramientas permiten mantener un historial de código y resolver conflictos durante la integración del código [9]:

- CVS (Concurrent Versions System) [10]: fue de las primeras herramientas que surgen para el control de versiones. Dejaron de añadirse nuevas funcionalidades en 2008. Utiliza un modelo de repositorio cliente-servidor (CVCS [11], Centralized Version Control System).
- Subversion (SVN) [12]: nació en el 2000 y es un proyecto de Apache. Se pueden utilizar softwares para trabajar con ella como SmartSVN y TortoiseSVN. Utiliza un modelo de repositorio cliente-servidor (CVCS [11], Centralized Version Control System).
- Git [13]: creado por Linus Torvalds en 2005, es una de las herramientas más flexibles y existen múltiples sistemas que permiten trabajar con ella (GitHub, GitLab, GitKraken…). Utiliza un modelo de repositorio distribuido (DVCS [11], Distributed Version Control System).
- Mercurial [14]: también se creó en 2005 y ofrece funcionalidades similares a las de Git. Utiliza un modelo de repositorio distribuido (DVCS [11], Distributed Version Control System).
- Perforce Helix Core [15]: lanzado en 1995, está diseñado para entornos de desarrollo a gran escala y para soportar grandes binarios, videos y contenido en general. Es centralizado (CVCS [11], Centralized Version Control System).

Se ha realizado una comparación entre estos sistemas de gestión de versiones teniendo en cuenta el modelo de repositorio, su coste, su funcionamiento online/offline, la popularidad entre los usuarios de VCS y si se permite su integación con herramientas de IC.

VCS	Modelo de repositorio	Licencia/Coste	Online/Offline	Popularidad (%) [16]	Integración IC
CVS	Cliente-Servidor y Centralizado.	GNU GPL / Gratis.	Muy limitado offline	0,8%	Ha dejado de mantenerse su integración con los servidores de IC.
SVN	Cliente-Servidor y Centralizado.	Apache / Gratis.	Muy limitado offline	13,5%	Se integra con Bamboo y Jenkins
Git	Distribuido.	GNU GPL / Gratis.	Mayoría de funcionalidades disponibles offline.	70,7%	Se integra con Bamboo, Jenkins y CircleCI.
Mercurial	Distribuido.	GNU GPL / Gratis.	Mayoría de funcionalidades disponibles offline.	13,5%	Se integra con Bamboo y con Jenkins mediante un *plugin*

Perforce	Cliente-Servidor y Centralizado.	De pago.	Limitado offline.	1,5%	Se integra con Bamboo y con Jenkins mediante un *plugin*.

Tabla 1. Comparativa VCSs

Si se desea comparar más aspectos de los VCS mencionados o de más que no se han descrito en este documento, se puede consultar una comparativa bastante extensa de aproximadamente 34 sistemas de control de versiones en Wikipedia [17]. Git es con diferencia la herramienta más popular entre los usuarios: es gratuita, se integra con la mayoría de sistemas de IC y existen muchas soluciones que facilitan su uso.

3.1.2 Compilación y generación de ejecutable

Como ya se ha mencionado, este proceso debe poder realizarse con un único comando y debe ser lo más rápido posible para que se cumplan los principios de la IC. Dependerá del lenguaje que se utilice, pero aquí se mencionan algunas de las herramientas más conocidas para su realización.

- Gcc [18]: compilador de C, C++, Objective- C, Fortran, Ada, Go y D. Forma parte del proyecto GNU.
- Make [19]: también de GNU, es una herramienta que controla la generación de ejecutables y otros ficheros a partir de los ficheros fuente.
- Ant [20]: librería Java y herramienta que gestiona el proceso de construcción de aplicaciones Java. Permite compilar, ensamblar, probar y ejecutar estas aplicaciones. Pertenece al proyecto Apache.
- Maven [21]: framework también parte del proyecto Apache, además de permitir la compilación, ensamblado, ejecución de pruebas y de una aplicación Java; permite realizar una mejor gestión de dependencias e implementa un "ciclo de vida", que consiste en que antes de ejecutar un objetivo, se ejecutan antes sus antecesores.
- Gradle [22]: permite la compilación, empaquetado, ejecución de pruebas y despliegue de aplicaciones Kotlin, Groovy, Scala, C/C++ y JavaScript entre otras. Se utiliza sobre todo en los proyectos Android.

Compiladores/ generadores de ejecutables	Lenguajes compatibles	Licencia/ Coste	Típicamente usado para…	Integración IC
Gcc	C, C++, Objective-C, Fortran, Ada, Go y D.	GNU / Gratis.	C y C++.	-
Make	C, C++, Pascal, Fortran...	GNU / Gratis.	C y C++.	-
Ant	Java	Apache / Gratis.	Java	Se integra con Jenkins, Bamboo, CircleCI, TeamCity, Travis CI.
Maven	Java	Apache / Gratis.	Java	Se integra con Jenkins, Bamboo, CircleCI, TeamCity, Travis CI.
Gradle	Kotlin, Groovy, Scala, C, C++,	Apache / Gratis.	Kotlin (Android)	Se integra con Jenkins, CircleCI, TeamCity, Travis CI.

	Javascript…			

Tabla 2. Comparativa compiladores/generadores de ejecutables

3.1.3 Pruebas

Una de las bases de la IC es la realización de pruebas sobre el software. Como se indica en el manual que proporciona ISTQB, la entidad más reconocida dedicada a las certificaciones en calidad software [23], las pruebas se pueden categorizar según los siguientes criterios:

- Niveles de pruebas:
  - Pruebas de componente o unitarias: se centran en los componentes que se pueden probar por separado.
  - Pruebas de integración: se centran en la integración entre distintos componentes o sistemas.
  - Pruebas de sistema: se centran en el comportamiento de un sistema completo.
  - Pruebas de aceptación: también se centran en el comportamiento del sistema completo, pero pretende identificar si cumple con la especificación de requisitos.
- Tipos de pruebas:
  - Pruebas funcionales: evalúa las funcionalidades del sistema, lo que este hace.
  - Pruebas no funcionales: se centra en características como la usabilidad, eficiencia o seguridad.
  - Pruebas de caja blanca: considera la estructura interna o implementación (por ejemplo el código y la arquitectura).
  - Pruebas relacionadas con cambios o de regresión: cuando se produce algún cambio en el software se debe verificar que este funciona correctamente y que no ha afectado al resto de funcionalidades.

Por otra parte, las pruebas pueden ser automáticas o manuales.

En este estudio se van a proponer herramientas para realizar pruebas funcionales unitarias y un plan de pruebas de regresión que permitan validar las funcionalidades básicas de la aplicación ante cualquier cambio.

3.1.3.1 Pruebas unitarias

Para la realización de este tipo de pruebas se suelen utilizar herramientas software que permiten escribir, construir y ejecutar pruebas unitarias, incluyendo muchas veces informes sobre el resultado de las pruebas [24]. Algunas de estas herramientas son las siguientes:

- xUnit y NUnit [25] [26]: herramientas para realizar pruebas unitarias de .NET [27].
- JUnit [28]: librerías para realizar pruebas unitarias de Java [29].
- CppUnit [30]: adaptación de JUnit para C++ [31].
- PyUnit [32]: para la creación de pruebas unitarias en Python [33].
- XMLUnit [34]: herramienta para realizar pruebas unitarias sobre el contenido XML [35].

No se realiza comparativa de estas herramientas ya que dependen del lenguaje que se utilice.

3.1.4 Análisis estático de código

Para complementar las pruebas propuestas, se comparan distintas herramientas de análisis estático de código con el propósito de detectar posibles problemas de seguridad y de mejorar la calidad del código. Algunas de estas herramientas son:

- bugscout [36]: plataforma para detección de vulnerabilidades de seguridad y análisis de calidad del código de aplicaciones. Integrable en el ciclo de IC.
- SonarQube [37]: revisor de código automático que detecta *bugs*, vulnerabilidades y *code smells* en el código. Permite su integración en los procesos de IC. Soporta lenguajes como Java, PHP, C, C++, C#, CSS, Go, JavaScript, SQL, Python… Es *opensource*.
- Kiuwan [38]: solución en la nube para detección de vulnerabilidades. También se integra en los procesos de IC.

Softwares de análisis estático	Lenguaje pruebas	Licencia/Coste	Tipos de pruebas	Integración IC
bugscout	Los 35 lenguajes más habituales.	De pago.	De seguridad y calidad.	Se integra con SonarQube, Maven, Ant, Bamboo, Jenkins, Travis CI, TeamCity
SonarQube	27 lenguajes (PHP, C, C++, C#, CSS, Go, JavaScript, SQL, Python…)	GNU / Gratis.	De seguridad y calidad.	Se integra con Jenkins.
Kiuwan	30 lenguajes (Java, Android, python, C++ JavaScript…)	De pago.	De seguridad y calidad. Especial enfoque en seguridad.	Se integra con Jenkins, Circle CI, Bamboo, TeamCity.

Tabla 3. Comparativa análisis estático de código

La herramienta gratutia más popular es SonarQube.

3.1.5 Servidores de IC

Para poder llevar a cabo la integración continua y automatizar el proceso, es indispensable la utilización de servidores de IC. Para ello existen varias opciones entre las que se encuentran las siguientes:

- Bamboo [39]: *pipeline* que permite la DC automatizando el proceso desde el código hasta el despliegue. Ofrece métricas de calidad.
- Jenkins [40]: servidor "de automatización". Es *opensource* y permite compilar, desplegar y en definitiva automatizar proyectos. Ofrece cientos de *plugins* que lo hacen muy flexible.
- CircleCI [41]: plataforma de IC y DC basada en la nube que permite automatizar el proceso.
- TeamCity [42]: servidor de IC y gestor de *builds*.
- Travis CI [43]: servidor de IC para construir *builds* y realizar pruebas.

Servidores IC	Tipo de *hosting*	Sistema Operativo	Licencia/Coste	Caso de uso [44]	*Plugins*
Bamboo	Instalación local o Bitbucket en la nube.	Windows, Linux, macOS, Solaris.	De pago.	Para integraciones de Atlassian.	Aproximadamente 200 *plugins*.
Jenkins	Instalación local o en la nube.	Windows, Linux, macOS y cualquiera Unix.	MIT y Creative Commons / Gratis.	Grandes proyectos.	Más de 1500 *plugins*.
CircleCI	En la nube.	Linux, iOS, Android.	De pago.	Desarrollo rápido y gran presupuesto.	Se integra con Artifactory, Gradle, Maven, Katalon…
TeamCity	Instalación local.	Windows, Linux, macOS, Solaris, FreeBSD…	De pago.	Para necesidades de empresas.	Aproximadamente 400.
Travis IC	Instalación local o en la nube.	Linux, macOS.	MIT / De pago.	Proyectos pequeños y startups.	En la documentación oficial solo aparecen aproximadamente 10 *plugins*.

Tabla 4. Comparativa de servidores de IC.

La herramienta gratuita que ofrece más posibilidades es Jenkins.

Aunque se usara cualquiera de estas herramientas hay que elegir una forma de realizar el despliegue. Se pueden utilizar scripts de despliegue, cuyo lenguaje dependerá de las máquinas utilizadas para los servidores de IC y se puede contemplar la opción de utilizar otras herramientas de gestión de configuración como Ansible [45], que permite gestionar las máquinas en las que se va a desplegar y realizar el propio despliegue.

3.1.6 Gestión de configuración

También denominadas SCM (Software Configuration Management) [46], son herramientas para la gestión de cambios en componentes software. Incluyen aspectos técnicos y comunicaciones. Permiten realizar los despliegues de forma más rápida y fácilmente escalable así como gestionar la infraestructura como si se tratara de código [47]. Se considera una herramienta indispensable dentro de DevOps [48].

- Puppet [49]: incluye su propio lenguaje para describir la configuración del sistema. Permite definir, probar y desplegar cambios utilizando la infraestructura como si se tratara de código y ofrece un cuadro de mandos para gestionar el inventario y hacer un seguimiento del mismo.
- Chef [50]: se usa para transformar la infraestructura en código mediante "recetas" que definen la infraestructura y dependencia de sistemas. Automatiza la gestión de infraestructura.
- Ansible [45]: modela la infraestructura describiendo como se relacionan los sistemas en Playbooks [51]. La diferencia de Ansible con otros SCM es que no utiliza agentes.

- CFEngine [52]: unifica la gestión de servidores, facilita cumplir con os niveles de servicio. Su primera versión se lanzó en 1993.

Gestores de configuración	Lenguaje de configuración	Arquitectura	Licencia/Coste	Integraciones
Puppet	Ruby, Puppet DSL.	Master-Agente.	Apache/GPL (depende de la versión). La versión Puppet open source es gratis.	Jenkins, GitLab, GitHub, CircleCI.
Chef	Ruby DSL.	Master-Agente.	Apache 2.0. La versión open source es gratis.	Jenkins, GitLab, Jenkins, git, JFrog.
Ansible	Python, YAML.	Sin agente. Solo nodo master.	GPLv3+. Uso de CLI gratis sin límites.	Github, GitLab, Jenkins, Travis CI, TeamCity, GitLab, GitHub… [53]
CFEngine	CFEngine DSL	Master-Agente.	GPLv3. Tiene una versión gratis has 25 nodos.	No ofrece documentación.

Tabla 5. Comparativa de servidores de gestores de configuración.

Existe una comparación más extensa en Wikipedia [54].

De acuerdo a las estadísticas [55] en los últimos años la mayoría de usuarios se ha decantado por Ansible debido a que es la herramienta más fácil de usar y a que no necesita agentes [56].

3.1.7 Gestión de repositorios

Para el almacenamiento de los paquetes generados al compilar una aplicación y para la gestión de las dependencias conviene la utilización de un gestor de repositorios [57]. También se puede utilizar para almacenamiento de imágenes de Dockers. Esto ofrece ventajas como: mejorar el tiempo de generación de binarios, reducir la dependencia de repositorios remotos y servir como una única fuente de descarga. Algunos gestores de repositorios remotos son los siguientes:

- Artifactory [58]: proyecto que se define en su web como el gestor de repositorios más avanzado del mundo. La versión *opensource* solo soporta paquetes generados por Maven, Gradle e Ivy.
- Nexus [59]: gestiona binarios y artefactos generados por las *builds*. Compatible con: eclipse, IntelliJ, Jenkins, Puppet, Docker…
- Archiva [60]: provee gestión de acceso, almacenamiento de artefactos generados por builds, indexación y navegación, informes de uso… Pertenece a Apache.

Gestores de repositorios	Tipos de archivos	Licencia/Coste	Integración IC
Artifactory	Generados por	De pago. Existe una versión gratuita.	Se integra con Jenkins, Bamboo, Circle CI, Travis CI, TeamCity…

	Maven, Gradle e Ivy.		
Nexus	APK, APT, Maven… [61]	De pago.	Se integra con Jenkins, Bamboo, Circle CI…
Archiva	Maven.	Apache / Gratis.	En la documentación no se habla de integraciones explícitamente.

Tabla 6. Comparativa análisis estático de código

A la hora de la integración, Artifactory ofrece muchas más posibilidades que Nexus [62].

4 HERRAMIENTAS Y ENTORNO SELECCIONADO

Para seleccionar qué herramientas se van a utilizar para la implementación del proceso de IC es necesario tener en cuenta el lenguaje de la aplicación que se vaya incorporar al proceso y el DBMS que se utilizará. También se elegirán herramientas gratuitas. Partiendo de la base de que se tratará de una aplicación Java y el DBMS será MySQL, las herramientas seleccionadas serán las que se detallan a continuación.

4.1 Herramientas seleccionadas

4.1.1 Servidor IC

Se selecciona el servidor Jenkins debido a que es el referente entre los gratuitos. Existen más servidores, pero la "CD Foundation" define Jenkins como el servidor de automatización *opensource* líder [63].

Puede utilizarse como un simple servidor de IC o como una herramienta de DC por lo que ofrece la posibilidad de realizar una implementación de DC en este proyecto.

Es una aplicación basada en Java y puede ser instalada en Windows, Linux, macOS y otros sistemas Unix.

Además, posee una muy intuitiva interfaz web desde la que configurar los cientos de *plugins* que desarrolla su amplia comunidad y que permiten integrarlo con casi cualquier herramienta utilizada en IC y DC. Estos plugins permiten incorporar infinidad de funcionalidades al servidor. Se pueden consultar en el sitio web de Jenkins [64], donde también puede revisarse su documentación.

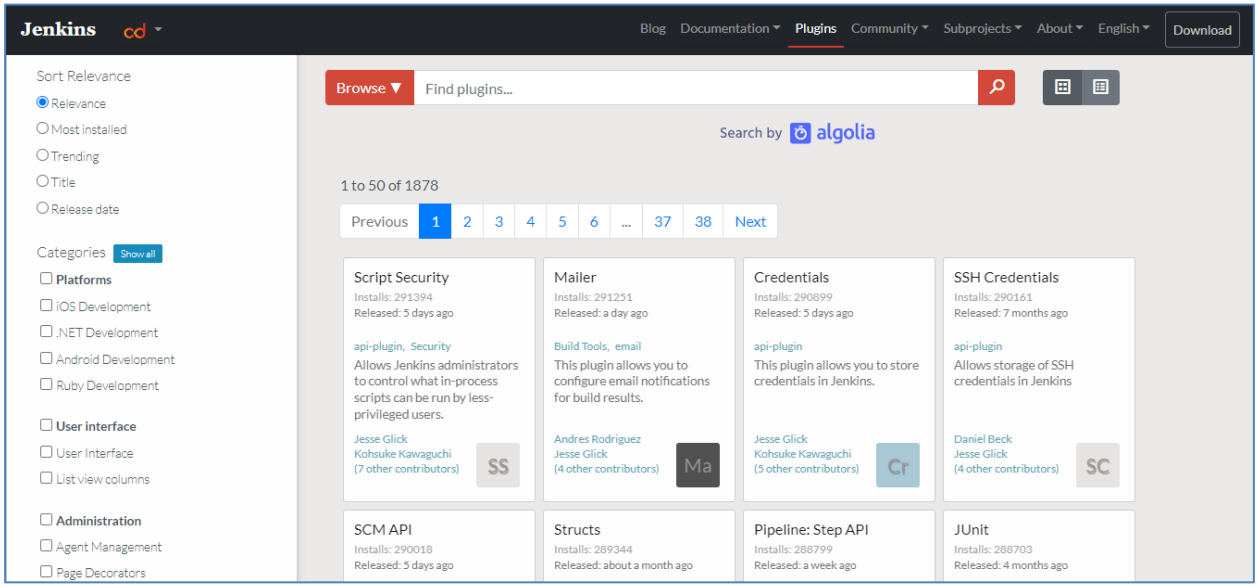

Imagen 1. Plugins de Jenkins.

4.1.2 VCS

El VCS elegido, basado en un modelo distribuido, es con diferencia el más popular entre los usuarios de este tipo de herramientas. Según su documentación oficial, lo más interesante de git y lo que lo destaca frente a otros gestores de versiones y código son las siguientes características [65]:

- *Branching* y *Merging*: este modelo permite tener varias ramas locales independientes unas de otras. Se pueden crear, fusionar y borrar en cuestión de segundos.

- Pequeño y rápido: la mayoría de las operaciones se realizan de forma local, lo que hace que sea mucho más rápido que los sistemas centralizados, al no tener que estar comunicándose con un servidor central constantemente. Además, está escrito en C, lo cual hace que sea mucho más rápido que otras herramientas escritas en lenguajes de más alto nivel.
- Distribuido: lo que significa que se hace un clonado del repositorio completo para trabajar con él. Esto permite que existan varias copias de seguridad y que puedan existir distintos flujos de trabajo.
- Aseguramiento de datos: asegura la integridad criptográfica de cada cambio en el proyecto.
- Área de *staging*: ofrece un área intermedia donde revisar los cambios antes de compartirlos.
- Es gratis y *opensource*.

Por último, es integrable con Jenkins y ofrece varios proveedores de *hosting* para alojar el código de forma gratuita.

4.1.3 Compilación y generación de ejecutable

Para la compilación, al tratarse de una aplicación Java, quedaban pocas opciones.

Se utiliza Maven porque es gratuito, compatible con Java e integrable con Jenkins y ofrece la ventaja de la implementación de un ciclo de vida [66] frente a Ant.

El concepto de ciclo de vida consiste en una definición del proceso de construcción y distribución de un artefacto que por defecto incluye las siguientes fases:

- validate: valida que el proyecto es correcto y que toda la información necesaria se encuentra disponible.
- Compile: compila el código fuente del proyecto.
- test: ejecuta las pruebas sobre el código fuente compilado utilizando un *framework* de pruebas unitarias disponible. Estas pruebas no necesitan que el código esté empaquetado o desplegado.
- package: empaqueta el código fuente en un archivo con el formato adecuado; por ejemplo, WAR.
- verify: ejecuta cualquier validación necesaria para asegurar que se cumplen los criterios de calidad.
- install: instala el paquete en el repositorio local para utilizarlo como una dependencia en otros proyectos.
- Deploy: copia el paquete final al repositorio remoto para compartirlo con otros proyectos o desarrolladores.

Al ejecutar una de las fases, se ejecutan las anteriores. Este ciclo de vida permite una generación del ejecutable ordenada y estándar para los proyectos Java.

Por otra parte, Maven ofrece una funcionalidad de gestión de dependencias [67] ampliamente utilizada en proyectos Java.

4.1.4 Pruebas unitarias

La herramienta por excelencia para la realización de pruebas unitarias en Java es JUnit.

Entre sus ventajas destacan:

- Incluye métodos de aserción para comprobar que las pruebas devuelven los resultados que se esperan.
- Incluye ejecutores gráficos para varios entornos de desarrollo, entre ellos Eclipse.
- Permite combinar pruebas en *suites* de pruebas.
- Permite deshabilitar pruebas.
- Permite establecer tiempos máximos de ejecución.

Es opensource, se integra con Maven, permite repetir las pruebas todas las veces que se quiera y generar informes de resultados.

4.1.5 Análisis estático de código

Para complementar las pruebas descritas, se elige para realizar un análisis de código estático la herramienta SonarQube, una aplicación que detecta de forma automática bugs, vulnerabilidades y *code smells* en el código. Indica el tiempo que se tarda en resolver cada problema y permite asignarlos a distintos usuarios. A demás de incluir reglas para más de 29 lenguajes, permite implementar perfiles de calidad que definen como evaluar el código.

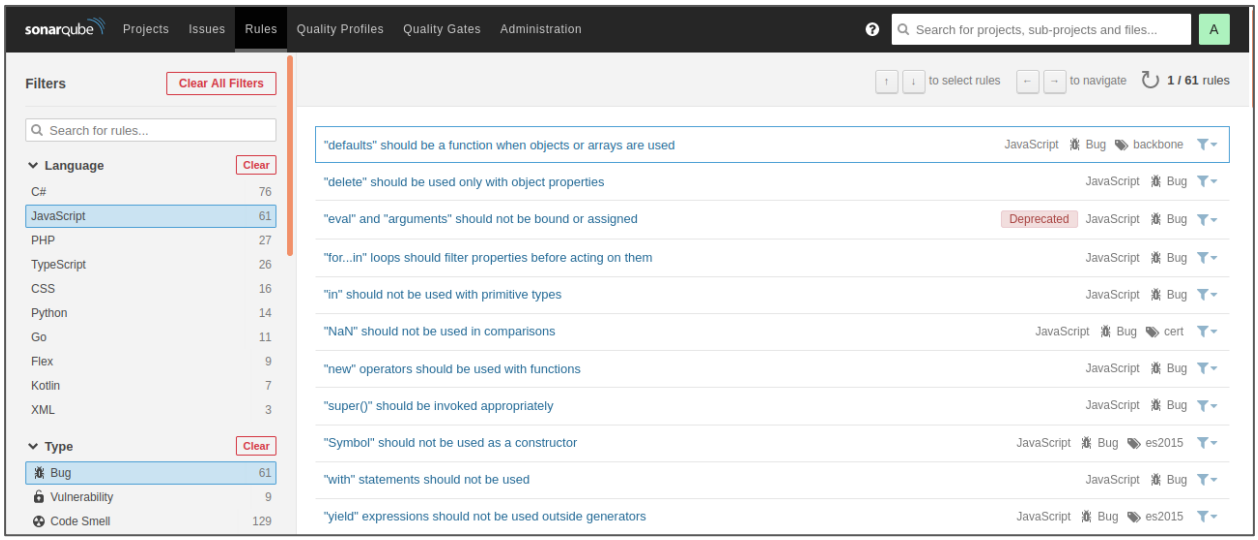

Imagen 2. Reglas de SonarQube.

Se trata de una herramienta gratuita, compatible con Java e integrable con Jenkins.

4.1.6 Gestión de repositorios

Para la gestión de los paquetes generados, se ha elegido la herramienta Artifactory que, como ya se ha comentado, se define en su web como el "gestor de repositorios más avanzado del mundo" [58]. Se trata de un gestor de repositorios que permite organizarlos según tres tipos:

- Locales: se utiliza para gestionar artefactos propios.
- Remotos: se utiliza para que Artifactory funcione como proxy almacenando artefactos externos que serán utilizados para las compilaciones de artefactos propios, reduciendo la dependencia de repositorios remotos y sirviendo como única fuente de descarga.
- Virtuales: sirve para organizar repositorios locales y remotos en un único repositorio.

Por lo tanto los artefactos que se generen se gestionaran en el repositorio local por defecto.

Existe una versión gratuita que ofrece muchas facilidades y se integra con Jenkins y Maven.

4.1.7 Gestión de configuración y despliegue

Para la gestión de la configuración de servidores y el despliegue de la aplicación, se ha utilizado Ansible.

Esta herramienta permite que se definan los servidores que se utilizan para la base de datos y despliegue, así

como ejecutar los comandos y despliegues necesarios en los servidores de aplicaciones con gran facilidad y sin necesidad de realizar ninguna instalación en estas máquinas.

No se necesita gran destreza como desarrollador ya que permite escribir instrucciones muy intuitivas con las que administrar la configuración y desplegar las aplicaciones.

Como ya se ha mencionado es gratuita, fácil de usar, no necesita agentes y es con diferencia la más popular en los últimos años.

4.2 Entorno seleccionado

La aplicación se desarrollará en una máquina Windows 10 con Eclipse [68], entorno de desarrollo Java integrable con Git, Maven, JUnit y Sonar. Inicialmente se desplegará en un servidor Tomcat [69] local para realizar las pruebas pertinentes en el entorno de desarrollo.

En resumen, se desarrollará una aplicación Java de ejemplo cuyo código se gestiona con Git, su DMBS será MySQL. Se elaborará un plan de pruebas funcionales y se generarán pruebas unitarias con JUnit y de código estático con SonarQube. Se realizará la implementación de una cadena Jenkins para automatizar su compilación con Maven, ejecución de pruebas y despliegue en un servidor de aplicaciones Tomcat. Los archivos WAR [70] generados se subirán a Artifactory.

Por último las herramientas Maven, Git, Jenkins, SonarQube y Ansible se instalarán en una máquina virtual local Ubuntu 18.04. El servidor de aplicaciones Tomcat 9.0.55 y el servidor de base de datos MySQL se instalarán en MVs en Google Cloud [71], herramienta que ofrece un periodo gratuito de prueba para crear máquinas de este tipo. El sistema operativo de la máquina virtual para el despliegue será Debian [72], que es uno de los sistemas operativos compatibles con el proyecto que ofrece Compute Engine [73]. Para la base de datos se utilizará un SQL Server de Google Cloud que consiste en un Microsoft SQL Server en el que se utiliza MySQL 8.0 [74].

En el anexo 8.6 se incluye un diagrama que describe la arquitectura y tecnologías utilizadas

5 IMPLEMENTACIÓN

Para documentar la implementación realizada se ha optado por describir primero las distintas herramientas utilizadas para más tarde describir la configuración realizada para su incorporación al proceso de Integración Continua.

5.1 Jenkins

Como se ha comentado anteriormente, Jenkins es el servidor de automatización *opensource* líder y el que ha sido seleccionado para la implementación que se va a realizar. Se configura mediante su interfaz web y cuenta con cientos de *plugins* que permiten integrarlo con casi cualquier herramienta utilizada en IC y DC.

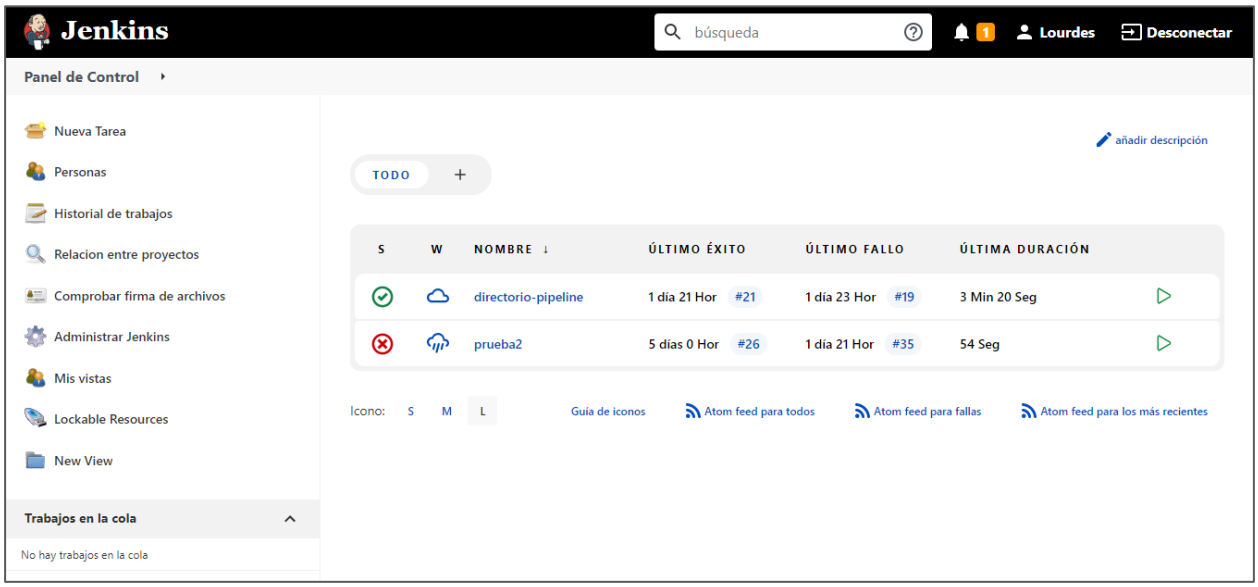

Imagen 3. Pantalla principal de Jenkins.

El concepto posiblemente más interesante de Jenkins es el de *Pipeline* [75], que es un conjunto de *plugins* que permiten implementar e integrar todos los procesos de DC. Estos *Pipelines* son código *DSL* o *Domain Specific Language*, que significa que están escritos en un lenguaje que se centra en un aspecto particular del software [76]; en este caso, existe un *DSL* para la implementación de *Pipelines*. Este código que describen las acciones a llevar a cabo y pueden definirse siguiendo dos tipos de sintaxis: *Declarative Pipeline* o *Scripted Pipeline* [77]. En este caso se utilizará el *Declarative Pipeline*, ya que es la sintaxis que permite definir el *pipeline* de una forma más estructurada.

El *Pipeline* se puede escribir directamente en Jenkins, pero se recomienda definirlo en un fichero Jenkinsfile [78] y almacenar el mismo en un VCS. Se puede consultar el código del pipeline utilizado en este proyecto en el anexo 8.2.

Para la instalación de la versión 2.319 de Jenkins en la máquina Ubuntu, se ha seguido la documentación oficial de Jenkins [79].

Los pasos a seguir para la **instalación** han sido los siguientes:

- Instalación de Java 11 desde la línea de comandos mediante la ejecución de

```
sudo apt-get install openjdk-11-jdk
```

- Importación de las GPG Keys de última versión mediante la ejecución de los comandos:

```
wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add
sudo sh -c 'echo deb https://pkg.jenkins.io/debian binary/ > \
/etc/apt/sources.list.d/jenkins.list'
```

- Instalación de Jenkins mediante la ejecución de:

```
sudo apt-get install jenkins
```

Una vez instalado se ha **registrado e iniciado como servicio** mediante la ejecución de los comandos:

```
sudo systemctl daemon-reload
sudo systemctl start jenkins
```

Por último se ha **configurado** realizando las siguientes acciones:

- Creación de usuario y contraseña siguiendo los pasos que se indican por pantalla tras la instalación.
- Actualización de idioma a español instalando el *plugin* "Locale Plugin" [80], accediendo a "Administrar Jenkins>Configurar el sistema" y estableciendo el campo "Default Language" a "es"
- Actualización de la zona horaria accediendo a "Administración>Gestión de usuarios", pulsando sobre el icono de configuración y estableciendo el campo "Time Zone" a "Europe/Madrid".

5.2 Git

Para trabajar con Git se ha optado por utilizar GitHub [81], un proveedor de *hosting* para el desarrollo de software y la gestión de sus versiones que utiliza Git. Ofrece muchas más funcionalidades y una interfaz web que permite la gestión del código de una forma bastante intuitiva.

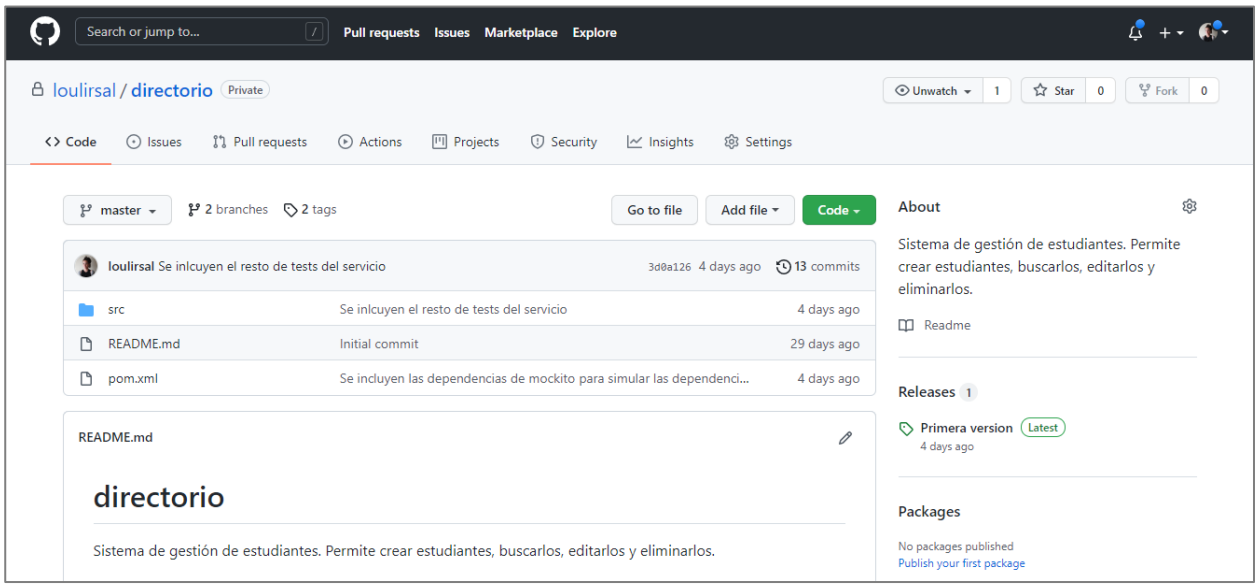

Imagen 4. Repositorio "directorio" en GitHub

También se utiliza Git Bash [82] para gestionar el código desde la máquina Windows donde se ha realizado el desarrollo.

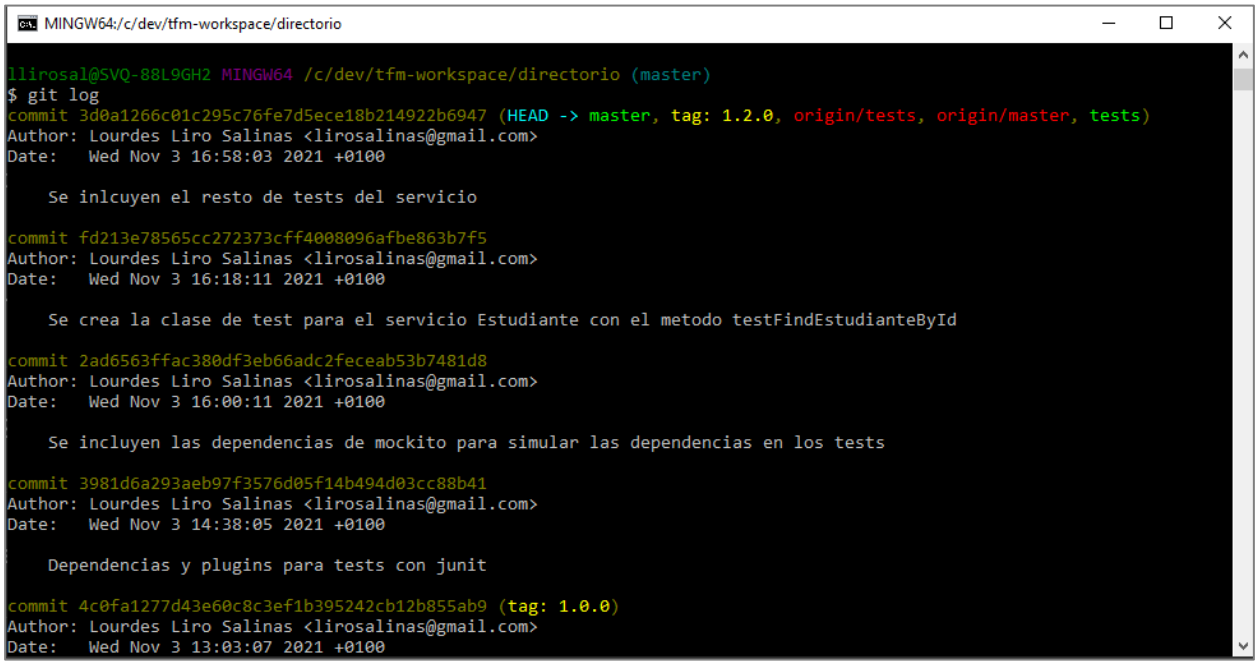

Imagen 5. Git Bash: línea de comandos Git para Windows.

Por último, se ha instalado un *plugin* de eclipse EGit [83] que permite gestionar información relacionada con Git desde el framework de desarrollo Eclipse.

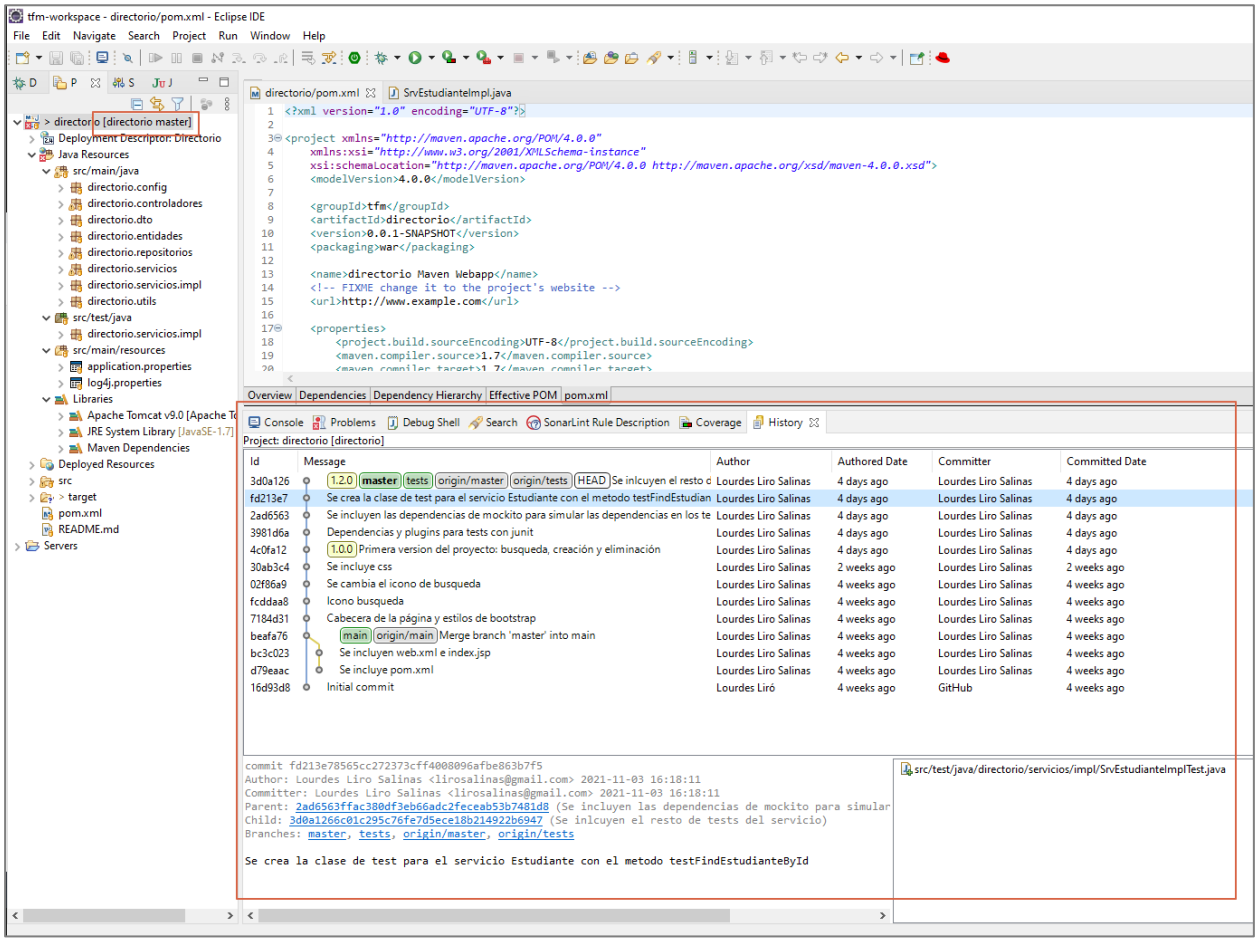

Imagen 6. EGit: plugin de git para Eclipse.

A continuación se indican los pasos seguidos para la creación del proyecto y su sincronización en la máquina local para permitir el desarrollo:

1. En primer lugar se crea el nuevo repositorio "directorio" desde GitHub (https://github.com/loulirsal/directorio).
2. Se genera una clave ssh para poder leer y escribir en el repositorio y se configura también en GitHub [84].
3. Se configura git en local [85] utilizando Git Bash.
4. Se sincroniza el proyecto git en local utilizando EGit mediante "Team > Share Project > Git".

En cuanto a la gestión de ramas, se crea una rama a partir de *master* [86], que será la rama principal, para cada nueva funcionalidad y se realiza un *merge* a *master* cuando esta funcionalidad sea estable; siendo un merge una una fusión entre ambas ramas [87]. Una vez se haya concluido una versión del software que incorpore las funcionalidades correspondientes, se crea un *tag* en master que indique la versión del software de la que se trata.

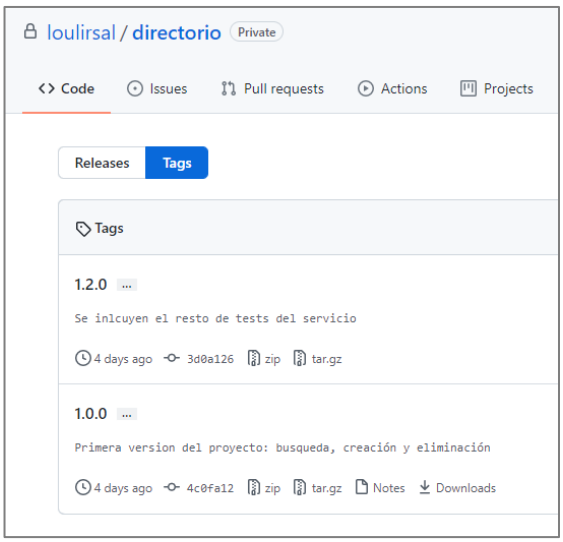

Imagen 7. Tags en GitHub

5.3 Aplicación Java

Se ha creado una aplicación web de prueba para poder mostrar el funcionamiento del proceso de integración continua. La aplicación consiste en un directorio de estudiantes que permite buscar, crear y eliminar estudiantes.

La pantalla inicial de la aplicación muestra el listado completo de estudiantes, un buscador y un botón para crear nuevos estudiantes. En cada fila se muestran los datos de los estudiantes (nombre, apellidos, fecha de nacimiento y correo) y un icono que permite eliminar el estudiante.

Imagen 8. Pantallas aplicación: pantalla inicial.

Al introducir parte de un nombre en el buscador y pulsar Intro, se mostrarán los resultados que coincidan en la tabla de estudiantes.

Imagen 9. Pantallas aplicación: búsqueda de estudiante.

Al pulsar sobre el botón “Nuevo estudiante” se mostrará una ventana modal que permitirá introducir los datos del estudiante.

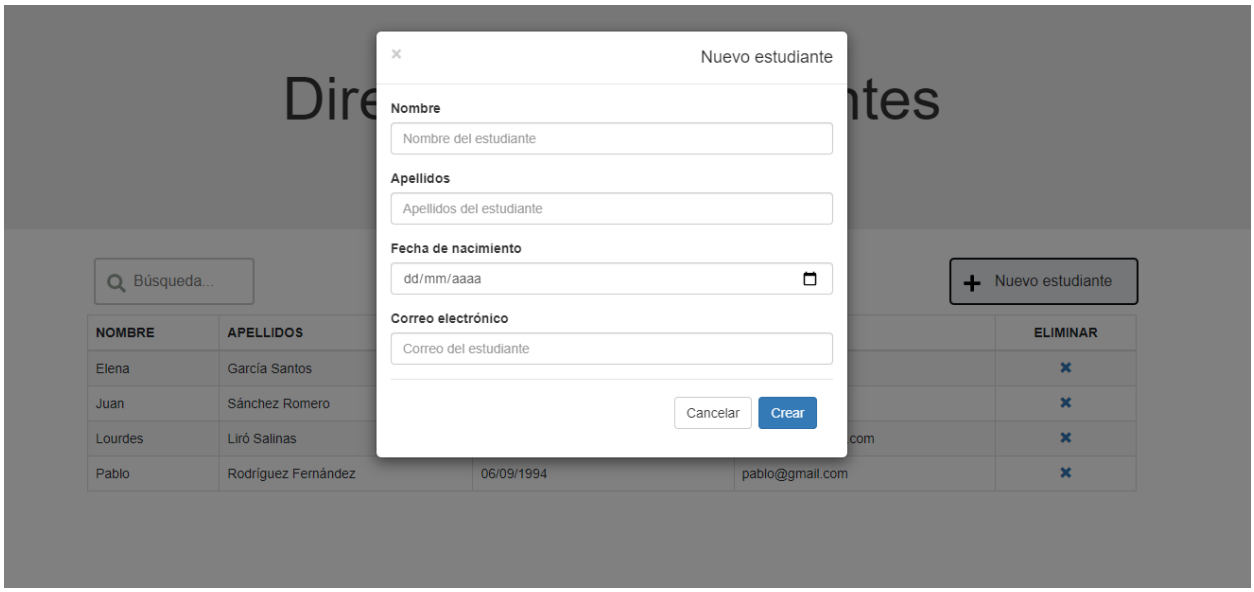

Imagen 10. Pantallas aplicación: modal "Nuevo estudiante".

Al completar los datos y pulsar el botón crear se procederá a crear el usuario. Si se crea, se mostrará un mensaje de éxito; en caso de que se produzca algún error durante su creación se mostrará un mensaje de error.

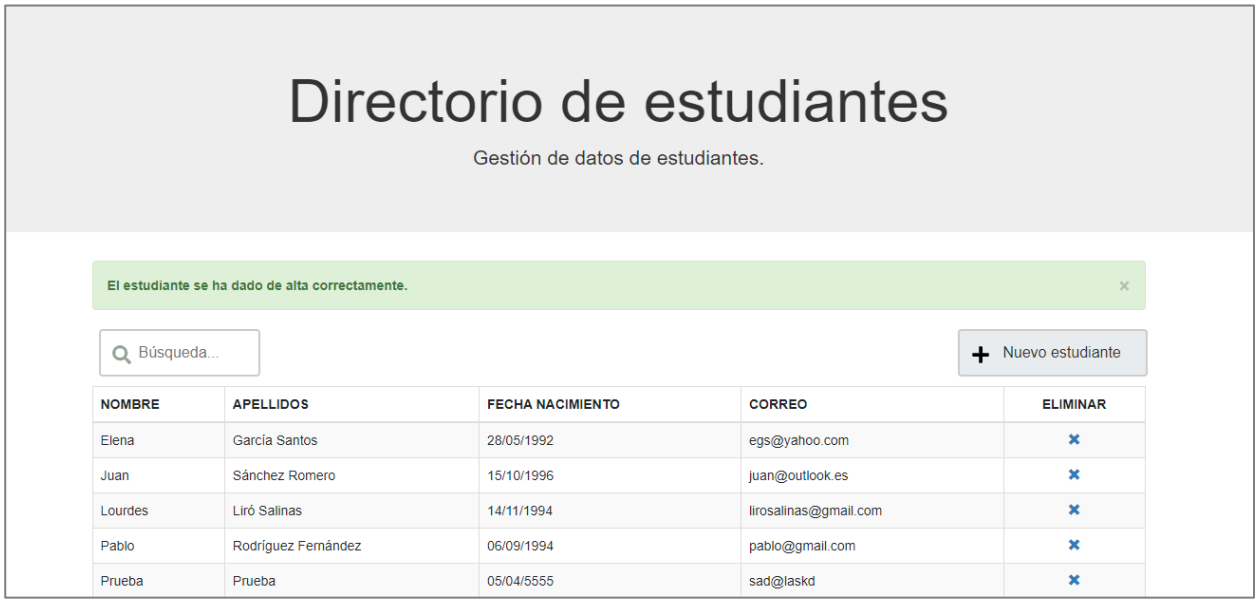

Imagen 11. Pantallas aplicación: mensaje de éxito al crear un estudiante.

Si se intenta crear un usuario con los mismos datos que un usuario ya existente, se mostrará el siguiente mensaje de error.

Imagen 12. Pantallas aplicación: mensaje de error en alta de estudiante.

Por último si se desea eliminar un usuario, se mostrará una modal de confirmación.

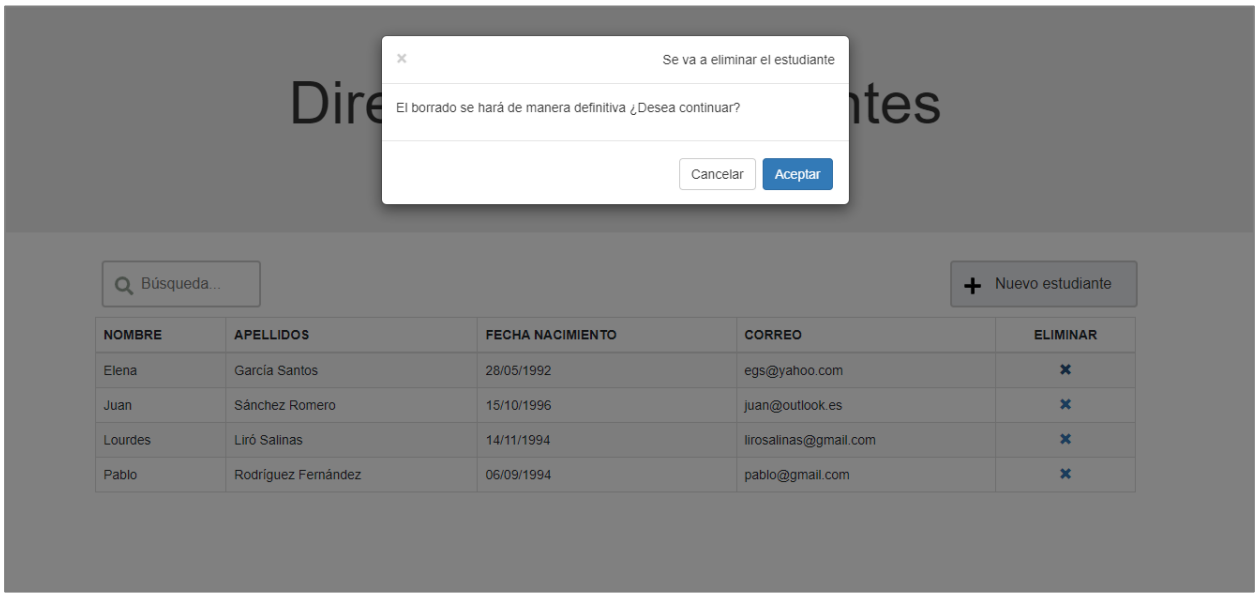

Imagen 13. Pantallas aplicación: mensaje de confirmación al eliminar.

Tras pulsar aceptar, se procederá a eliminar el usuario. Si se elimina, se mostrará un mensaje de éxito; en caso de que se produzca algún error durante su borrado se mostrará un mensaje de error.

Imagen 14. Pantallas aplicación: mensaje de éxito al eliminar.

Para el modelo de datos se ha utilizado un servidor MySQL 5.7.36 for Linux (x86_64) [88] y para realizar pruebas se ha instalado en la máquina virtual Ubuntu 18 utilizando el comando:

```
apt-get install mysql-server
```

Se ha creado la base de datos "directorio" y la tabla "estudiante" para almacenar los datos necesarios. Se puede consultar el contenido de los scripts utilizados para la creación y los creados para una posible marcha atrás en el anexo 8.

Para poder consultar la base de datos desde la máquina Windows donde se ha realizado el desarrollo de la aplicación web de prueba se han llevado a cabo los siguientes pasos:

- Se ha configurado las preferencias de red de la máquina virtual para que obtenga una IP accesible por la máquina Windows.

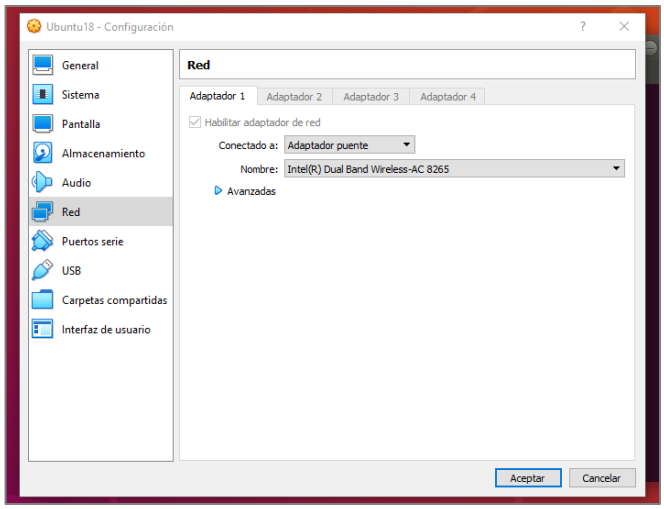

Imagen 15. BBDD: Adaptador de red MV Ubuntu.

- Se ha instalado el software MySQL Workbench 8.0 CE [89] y se ha configurado la conexión a la base de datos creada en la máquina virtual.

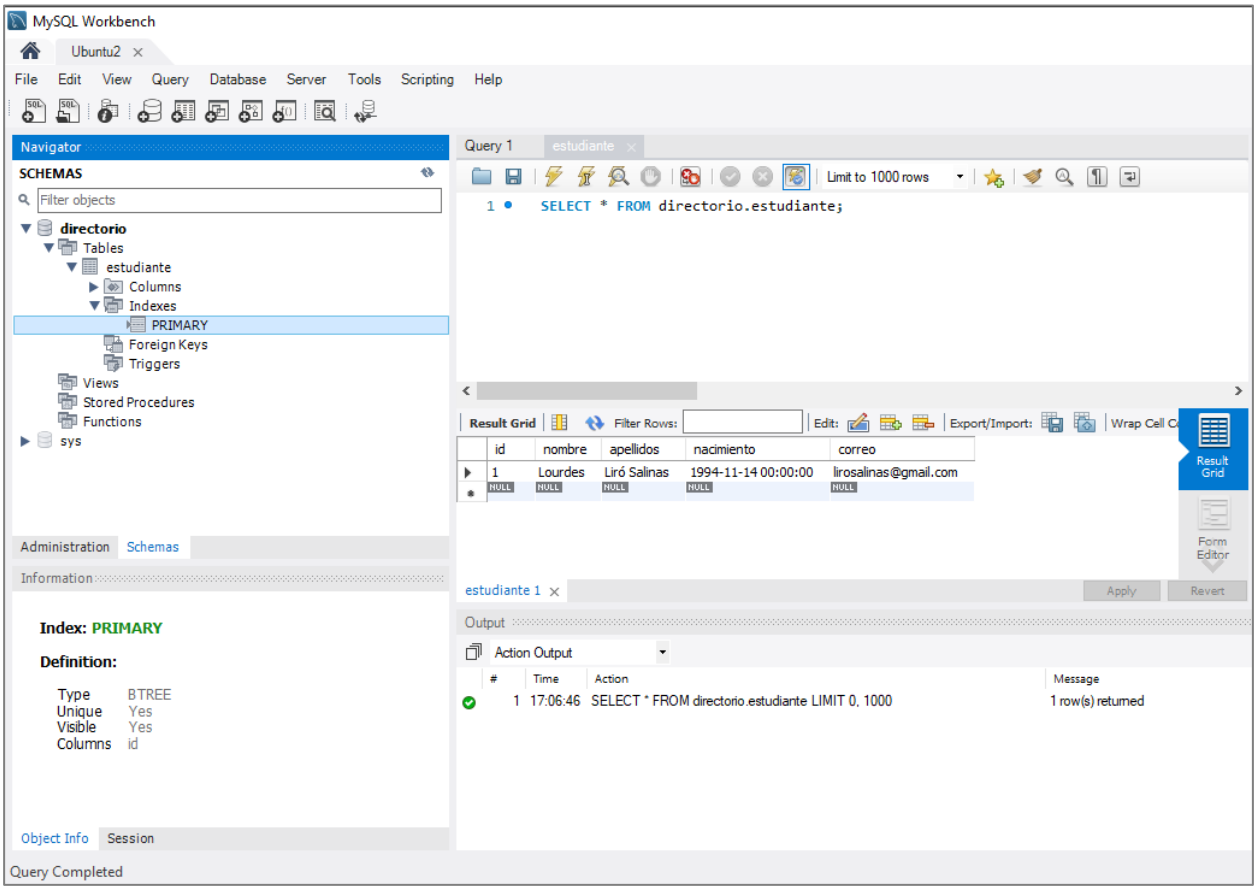

Imagen 16. BBDD: MySQL Workbench.

Para el desarrollo de la aplicación Java se ha creado un proyecto Maven en Eclipse Java Developers [90] y se han utilizado JDK 11, Bootstrap [91], Hibernate [92] y Spring [93] para facilitar el desarrollo.

Para su despliegue se ha configurado un servidor Tomcat 9 utilizando eclipse [69].

Una vez que se ha comprobado el correcto funcionamiento, se han creado la base de datos y el servidor Tomcat en máquinas de Google Cloud Platform [94], un servicio ofrecido por Google que permite crear y gestionar máquinas virtuales mediante una interfaz web y conexión ssh entre otras opciones.

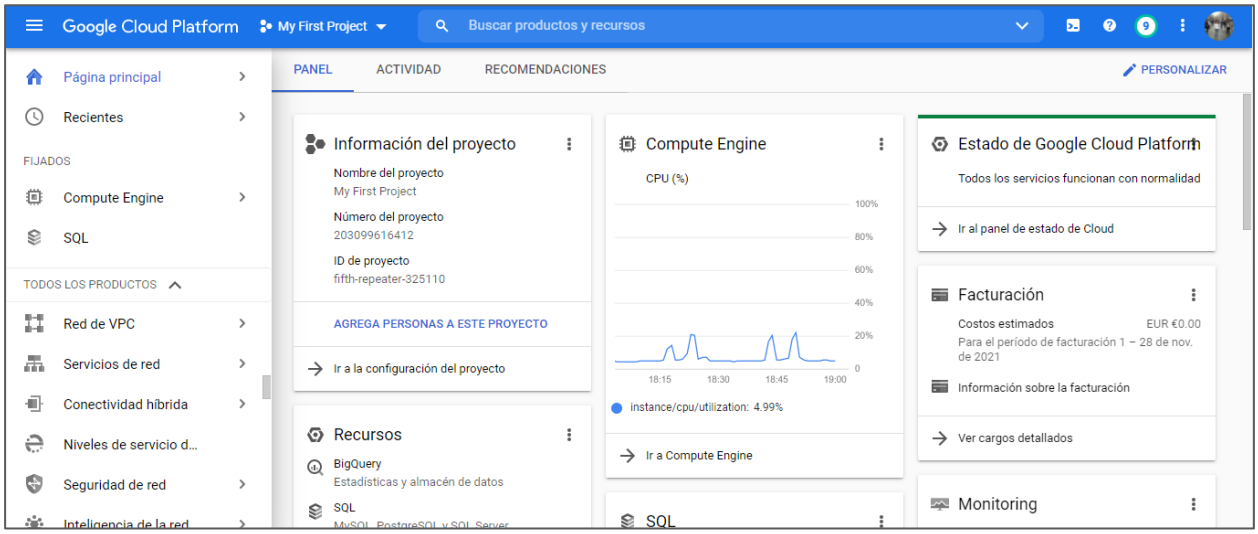

Imagen 17. Pantalla principal de Google Cloud Platform.

Se ha replicado la base de datos en un SQL Server con MySQL:

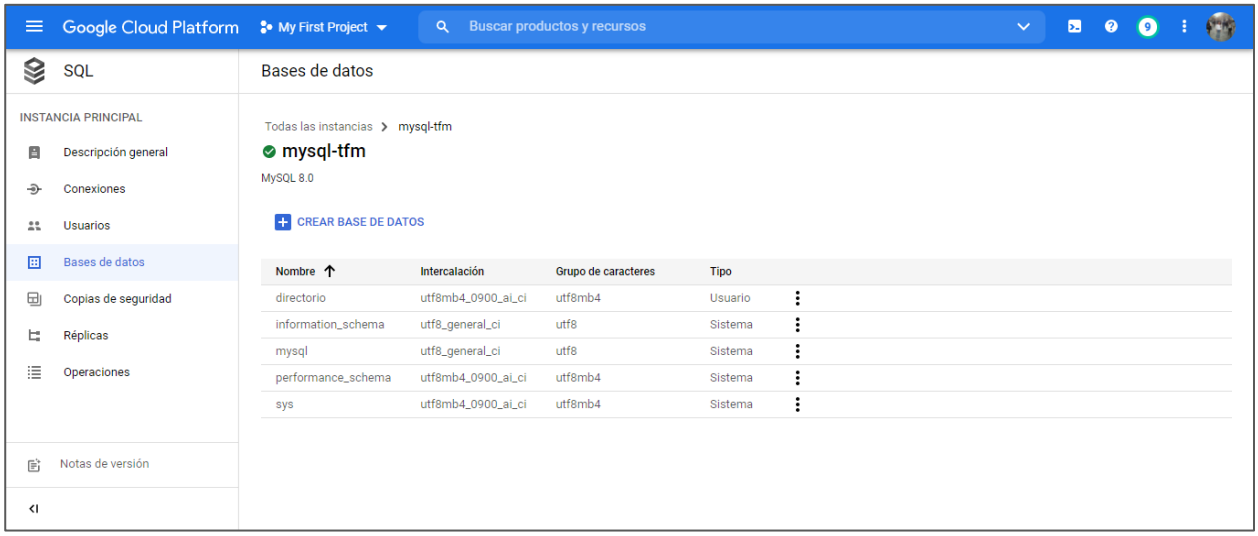

Imagen 18. Máquina SQL en Google Cloud Platform.

Y se ha creado el servidor Tomcat en una máquina Debian 9 utilizando la funcionalidad Compute Engine:

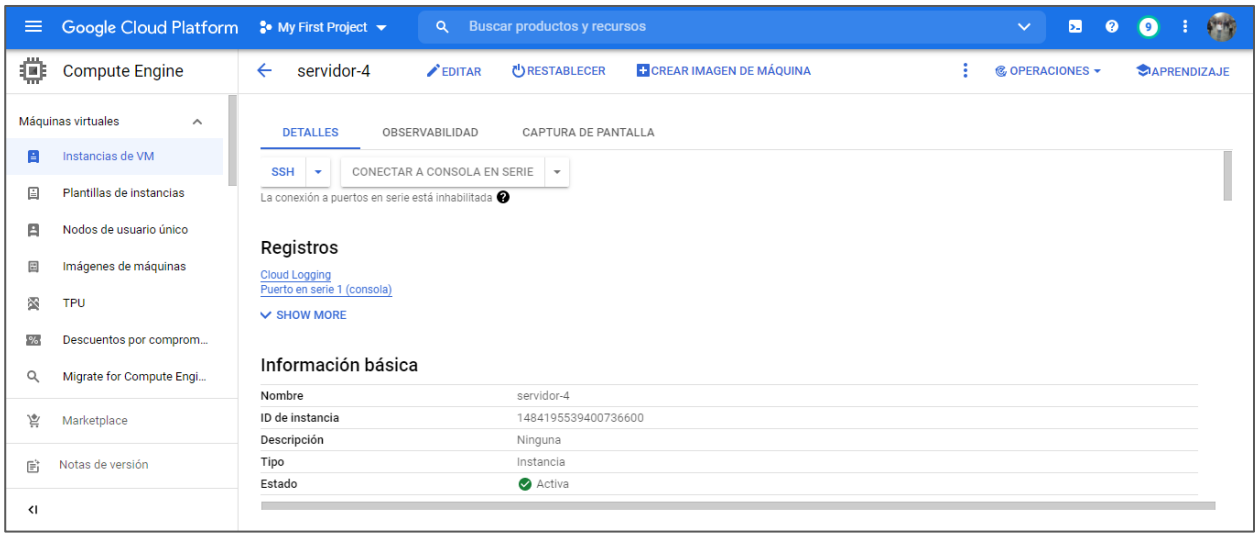

Imagen 19. Máquina Debian en Google Cloud Platform.

5.4 Compilación con Maven

Al tratarse de una aplicación Java, para la compilación y generación del WAR se ha utilizado Maven. Durante el desarrollo, se ha configurado la compilación desde Eclipse y se ha utilizado la versión 3.6.3.

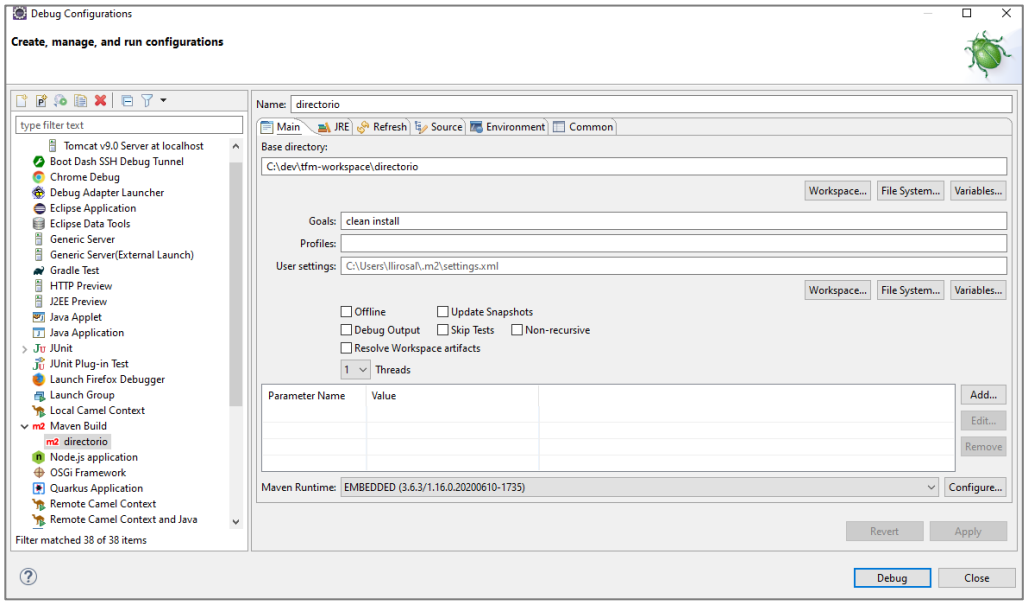

Imagen 20. Configuración de Maven en Eclipse.

Para la instalación en la máquina virtual donde se instala Jenkins, se ha seguido la documentación oficial de Maven [95] y se ha instalado la misma versión utilizada para el desarrollo siguiendo los siguientes pasos:

- Descarga mediante la ejecución de:

```
wget https://dlcdn.apache.org/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz
```

- Descompresión mediante la ejecución de:

```
tar xzvf apache-maven-3.6.3-bin.tar.gz
```

- Se mueve a /opt/apache-maven-3.6.3 mediante:

```
mv apache-maven-3.6.3 /opt/
```

- Se crea un enlace simbólico mediante:

```
ln -s /opt/apache-maven-3.6.3 /opt/maven
```

- Se crea la variable de entorno M2_HOME ejecutando:

```
echo 'export M2_HOME=/opt/maven' >> ~/.bashrc
```

- Se incluye en el PATH con:

```
echo 'export PATH=${M2_HOME}/bin:${PATH}' >> ~/.bashrc
```

5.5 Pruebas unitarias

Las primeras pruebas que se realizan sobre la aplicación son unas pruebas unitarias con el ya mencionado JUnit.

Para la implementación de los test se ha utilizado Mockito [96] para facilitar la realización de las pruebas y se ha realizado una clase de prueba sobre el servicio que permite realizar acciones sobre la entidad estudiante (listado, búsqueda, guardado y borrado). Se han implementado test para los métodos de esta clase.

Mediante la ejecución del siguiente comando se pueden ejecutar las pruebas creadas y obtener un informe de sus resultados:

```
mvn test
```

Además, utilizando Eclipse se puede comprobar lo que se denomina la cobertura, que es el porcentaje de código para el que se realizan pruebas.

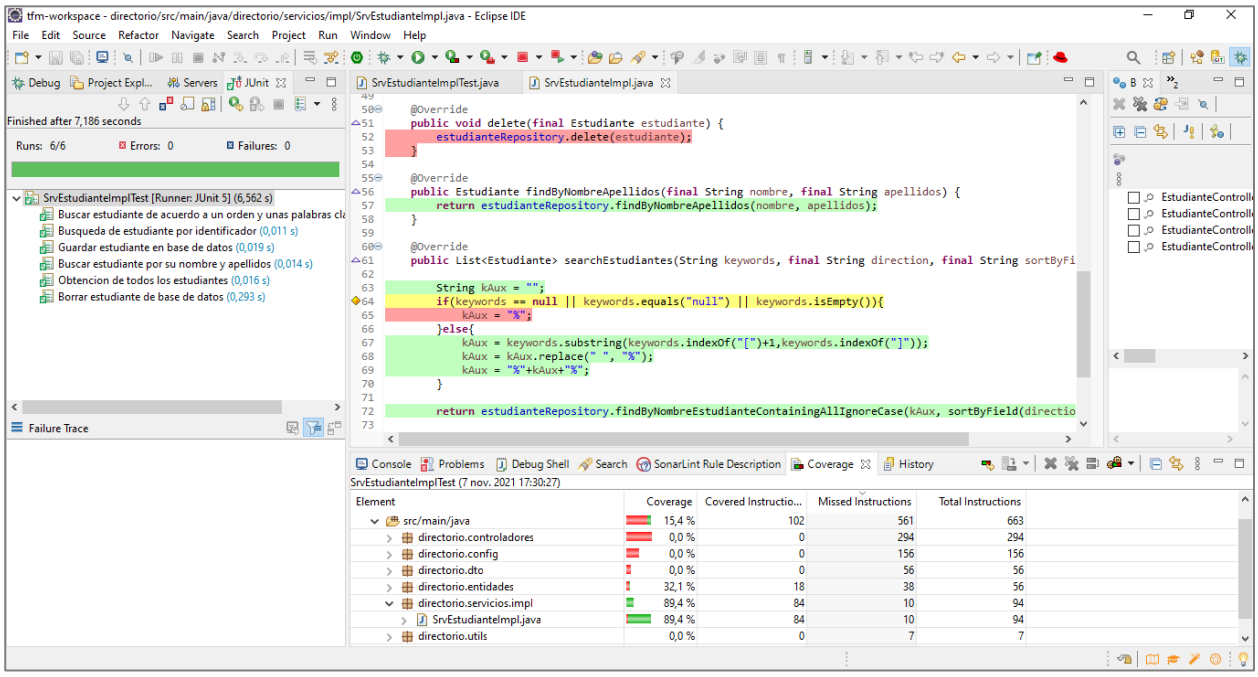

Imagen 21. Cobertura de test JUnit en Eclipse.

5.6 Plan de pruebas

Tras la creación de las pruebas unitarias, se propone un plan de pruebas funcional que se basa en varias pruebas de regresión para validar las funcionalidades básicas de la aplicación. Las pruebas de este plan se elaboran en base a la plantilla que proporciona el Marco de Desarrollo de la Junta de Andalucía [97] y son las siguientes:

Acceso a página de inicio	Código del CP	FUN-001
Descripción: Acceso a la página de inicio de la aplicación en la que se muestra el listado de estudiantes existentes.		
Prerrequisitos: N/A		
Pasos: 1. Acceder a la URL en la que se encuentra publicada la aplicación (por ejemplo: http://localhost:8080/directorio).		
Resultado esperado: Se muestra una tabla en la que aparecen los datos de los estudiantes que hayan sido registrados previamente.		

Creación de usuario	Código del CP	FUN-002
Descripción: Creación de un usuario en el directorio de estudiantes.		

Prerrequisitos:
- Ejecución de FUN-001
- Que no se haya creado previamente ningún estudiante con los mismos datos que los del usuario a crear

Pasos:
1. Pulsar el botón “Nuevo Estudiante”.
2. Completar el formulario con los siguientes datos:
- Nombre: “Nombre”
- Apellidos: “Apellido1 Apellido2”
- Fecha de nacimiento: “01/01/1111”
- Correo electrónico: “prueba@prueba.com”
3. Pulsar el botón “Crear”.

Resultado esperado:
El estudiante creado se muestra en la tabla y aparece el mensaje “El estudiante se ha dado de alta correctamente”.

Búsqueda de usuario	Código del CP	FUN-003

Descripción:
Búsqueda de un usuario en el directorio de estudiantes.

Prerrequisitos:
- Ejecución de FUN-002

Pasos:
1. Escribir “Nom” en la cajetilla de búsqueda.
2. Pulsar intro.

Resultado esperado:
En la tabla se muestran los estudiantes que incluyan la palabra “Nom” en su nombre; por lo tanto se debe mostrar el que se ha creado al ejecutar FUN-002.

Borrado de usuario	Código del CP	FUN-004

Descripción:
Borrado de un usuario en el directorio de estudiantes.

Prerrequisitos:
- Ejecución de FUN-002

Pasos:
1. Pulsar sobre el icono de borrado de la columna “ELIMINAR” para la fila en la que se encuentra el usuario creado en FUN-002.
2. Pulsar “Aceptar” cuando se muestre la ventana de confirmación de borrado.

Resultado esperado:
En la tabla se muestran los estudiantes ya no se muestra el estudiante creado en FUN-002 y aparece el mensaje “El estudiante se ha eliminado correctamente”.

5.7 Sonar

Para la instalación de SonarQube en la máquina Ubuntu se han seguido las instrucciones indicadas en la documentación oficial [98] y en otras fuentes [99]. Se ha utilizado la base de datos PostgreSQL, ya que es la que menos incompatibilidades puede presentar [100]. En concreto los pasos seguidos han sido los siguientes:

- Instalación de PostgreSQL:

```
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/ `lsb_release -cs`-pgdg main" >> /etc/apt/sources.list.d/pgdg.list'

wget -q https://www.postgresql.org/media/keys/ACCC4CF8.asc -O - | sudo apt-key add –

sudo apt-get -y install postgresql postgresql-contrib

sudo systemctl start postgresql

sudo systemctl enable postgresql

sudo passwd postgres #cambio de contraseña del usuario postgress
```

- Creación de usuario, base de datos y privilegios en PostgreSQL :

```
su – postgres

createuser sonar

psql

ALTER USER sonar WITH ENCRYPTED password 'my_strong_password';

CREATE DATABASE sonarqube OWNER sonar;

GRANT ALL PRIVILEGES ON DATABASE sonarqube to sonar;

\q

exit
```

- Instalación de SonarQube:

```
wget    https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-7.3.zip

sudo unzip sonarqube-7.3.zip -d /opt

sudo mv /opt/sonarqube-7.3 /opt/sonarqube

sudo groupadd sonar

sudo useradd -d /opt/sonarqube -g sonar sonar

sudo chown sonar:sonar /opt/sonarqube –R
```

- Modificación del host, puerto, url, usuario y contraseña del esquema de base de datos en el archivo sonar.properties:

```
sudo nano /opt/sonarqube/conf/sonar.properties
```

- Descomentado y edición de “RUN_AS_USER=sonar” en el archivo sonar.sh:

```
sudo nano /opt/sonarqube/bin/linux-x86-64/sonar.sh
```

- Creación de archivo de servicio systemd para arranque en etc/systemd/system/sonar.service con el contenido:

```
[Unit]
```

```
Description=SonarQube service
After=syslog.target network.target

[Service]
Type=forking

ExecStart=/opt/sonarqube/bin/linux-x86-64/sonar.sh start
ExecStop=/opt/sonarqube/bin/linux-x86-64/sonar.sh stop

User=sonar
Group=sonar
Restart=always

LimitNOFILE=65536
LimitNPROC=4096

[Install]
WantedBy=multi-user.target
```

- Arranque y habilitación del servicio:

```
sudo systemctl start sonar
sudo systemctl enable sonar
```

Tras ello se ha realizado la configuración inicial de sonar, indicando el tipo de proyecto que se analizará.

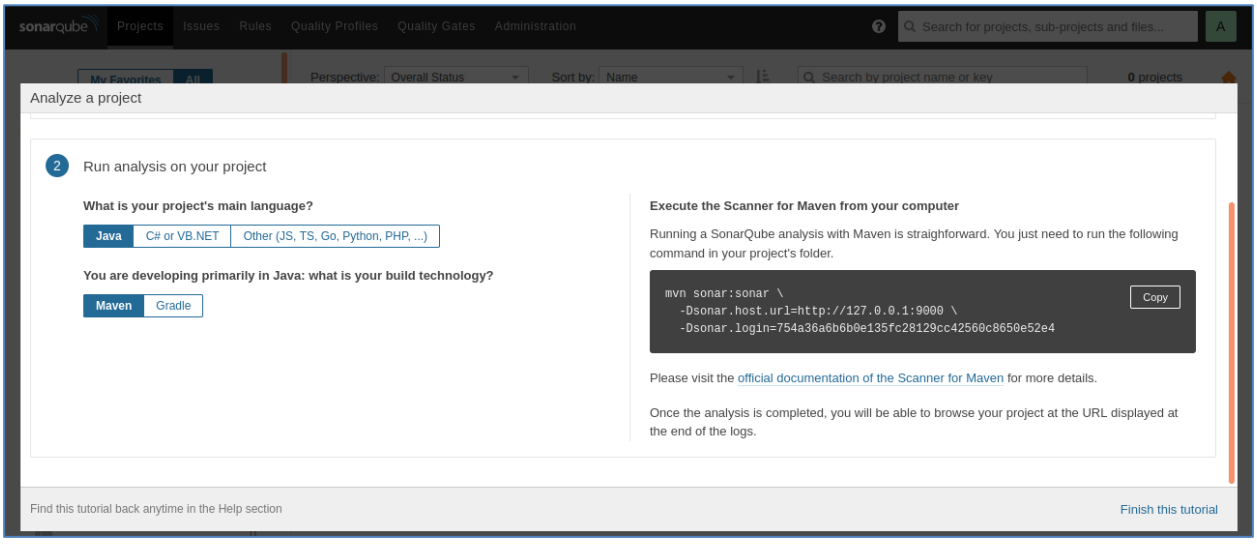

Imagen 22. Configuración inicial de SonarQube.

También se puede utilizar en Eclipse a través del *plugin* SonarLint [101].

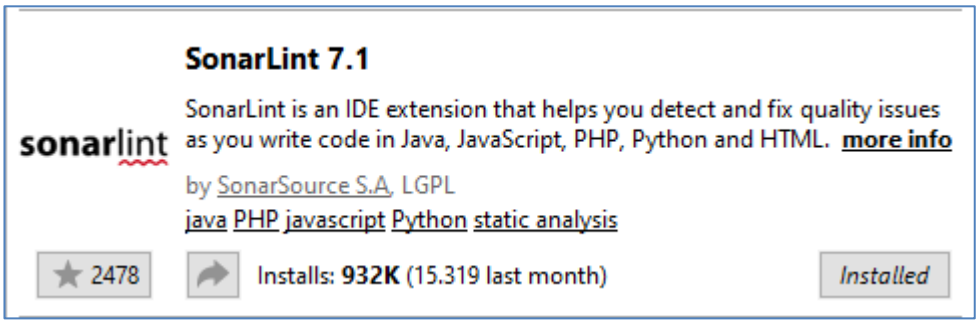

Imagen 23. SonarLint para Eclipse.

Este *plugin* indica en tiempo real si se está incumpliendo alguna regla de sonar al escribir el código:

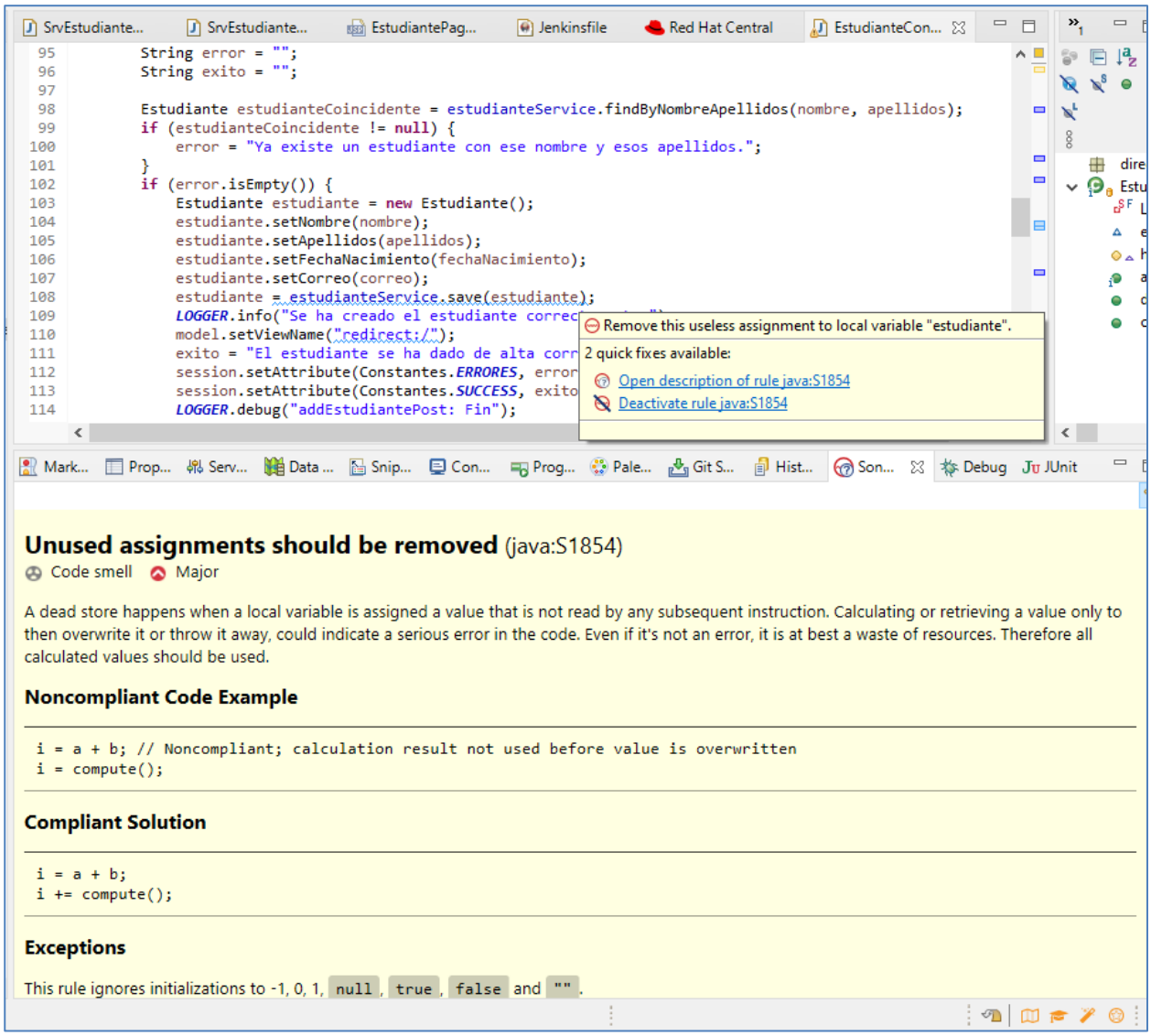

Imagen 24. Plugin SonarLint para Eclipse.

5.8 Artifactory

Para trabajar con Artifactory simplemente se ha creado una cuenta gratuita a través de la página oficial [102].

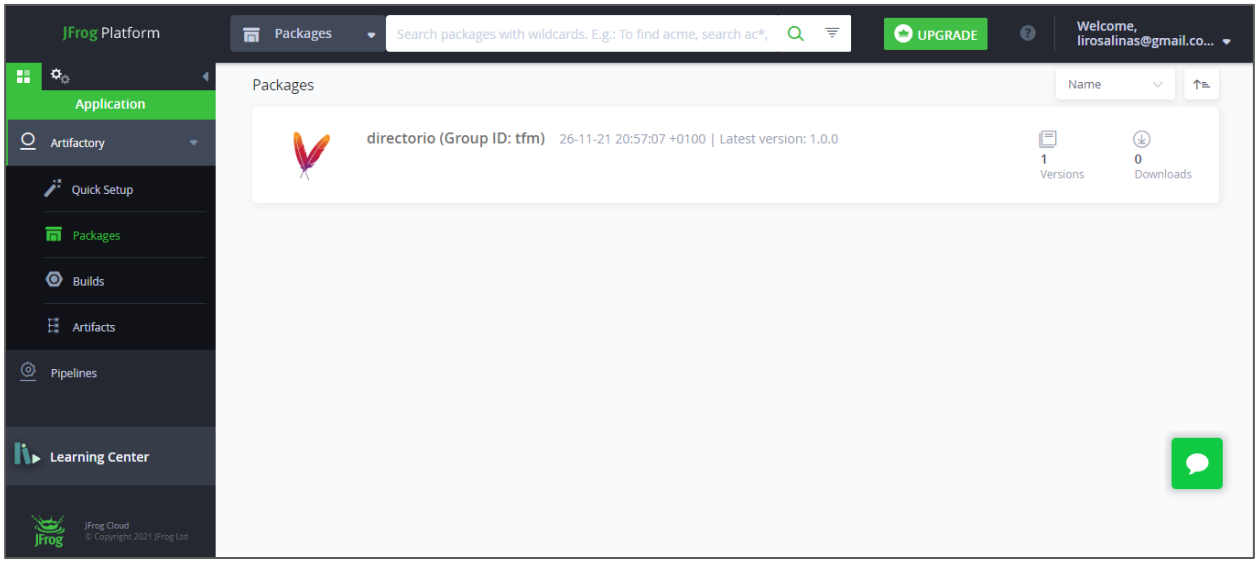

Imagen 25. Plugin SonarQube para Eclipse.

La URL en la que se ha creado es https://tfm.jfrog.io/ . En el apartado 5.10.5 se describe cómo se ha integrado con Jenkins para almacenar los artefactos generados cada vez que se crea uno.

5.9 Ansible

Se ha realizado la instalación de Ansible en la máquina Ubuntu siguiendo la documentación oficial [103], que consiste en ejecutar los siguientes comandos:

```
sudo apt update
sudo apt install software-properties-common
sudo add-apt-repository --yes --update ppa:ansible/Ansible
sudo apt install ansible
```

También se ha generado una clave ssh para poder ejecutar el despliegue en los servidores correspondientes:

```
ssh-keygen -t rsa
ssh-add ~/.ssh/id_rsa
```

Tras generarla se incorporará a los servidores para poder enviar las instrucciones de despliegue por ssh. Estos servidores se configuran en el archivo /etc/ansible/hosts:

```
## 192.168.100.10

# Ex 2: A collection of hosts belonging to the 'webservers' group

## [webservers]
## alpha.example.org
## beta.example.org
## 192.168.1.100
## 192.168.1.110

[servidores-prueba]
servidor-1
servidor-2

[servidor]
servidor-4

[bbdd]
mysql-bbdd
# If you have multiple hosts following a pattern you can specify
# them like this:

## www[001:006].example.com

# Ex 3: A collection of database servers in the 'dbservers' group
```

Imagen 26. Hosts definidos para Ansible.

En la imagen anterior se indican nombres de hosts y no direcciones IP porque estas han sido asociadas a los nombres indicados en el archivo /etc/hosts de la máquina.

5.10 Incorporación al proceso de IC

5.10.1 Git

Para realizar la integración de Jenkins con Git y así poder descargar el código en la máquina Ubuntu, lo primero que habría que hacer sería instalar Git en la máquina mediante la ejecución del comando:

```
apt-get install git
```

Una vez realizada la instalación, se debe reiniciar Jenkins y comprobar la instalación de Git accediendo a "Administrar Jenkins>Global Tool Configuration > Git > Git installations".

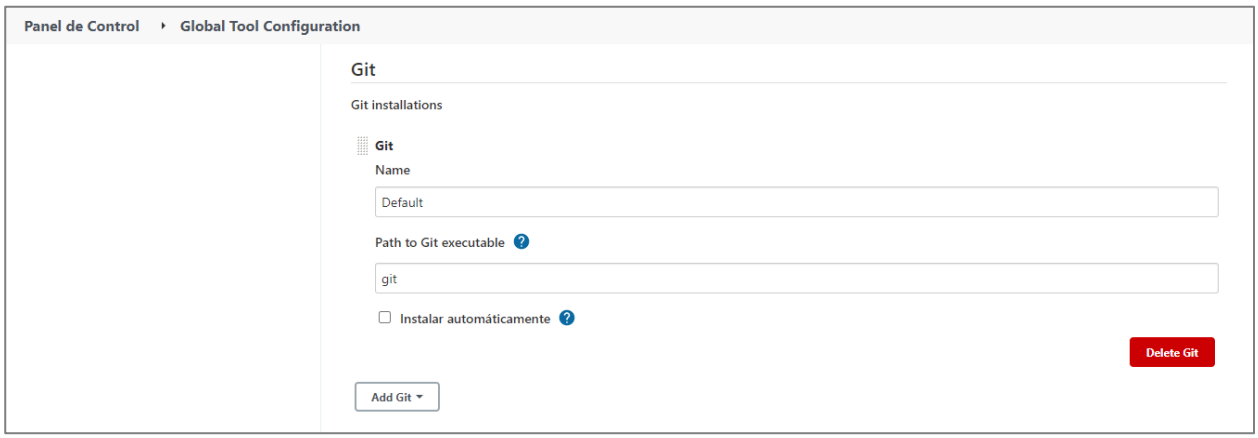

Imagen 27. Instalación de Git para Jenkins

Aquí se puede comprobar que Jenkins reconoce la instalación.

Para poder descargar el código también es necesario que se creen las credenciales, comenzando por la generación de una clave pública y privada. Para ello se genera y añade al agente ssh en la máquina en la que se encuentra instalado Jenkins y a GitHub [104]:

- Generación de clave:

```
ssh-keygen -t ed25519 -C "correo de acceso a Git"
```

- Se añade la clave al agente ssh:

```
ssh-add ~/.ssh/id_ed25519
```

- Para añadirlas a GitHub se accede a "Settings>SSH keys" y se incluye la clave pública (en la imagen se observa la clave creada para el desarrollo de código en la máquina local y para la descarga del código en la máquina de Jenkins).

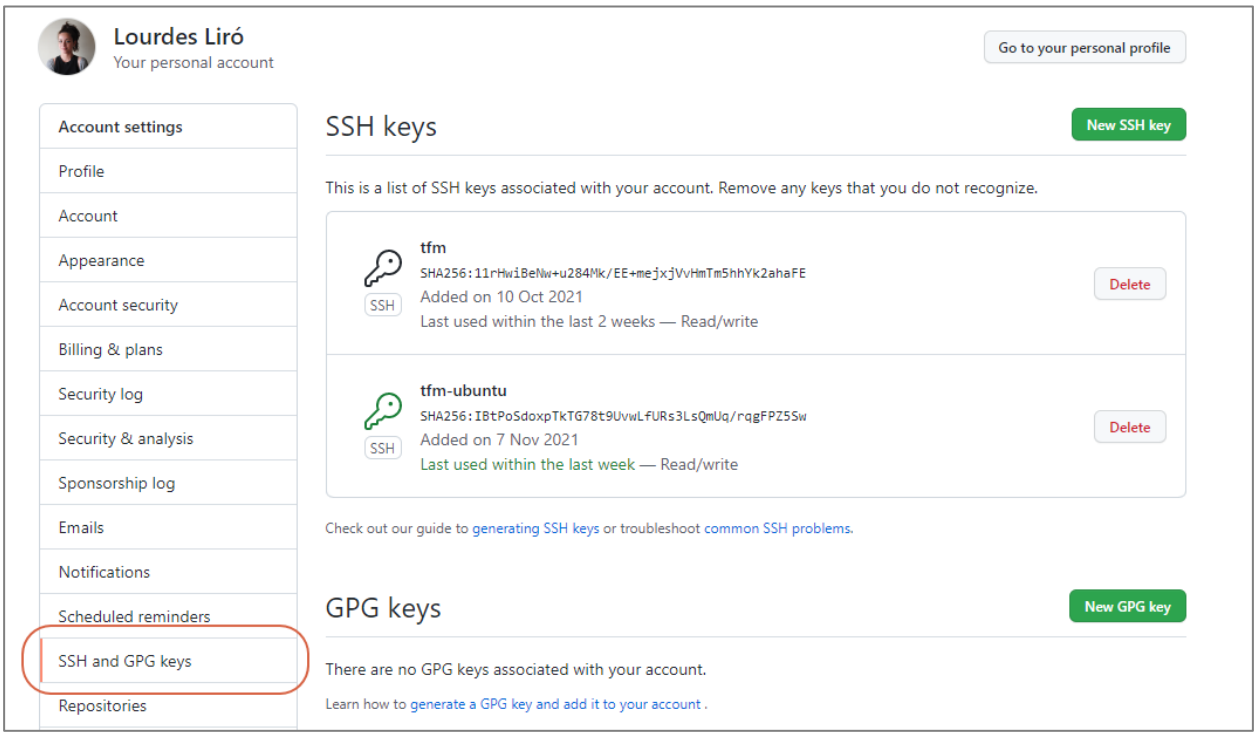

Imagen 28. Claves ssh en GitHub

- Por último se añaden las credenciales en Jenkins accediendo a "Administrar Jenkins>Manage Credentials". Se pulsa sobre "Add domain" y dentro del dominio "Global" se incluye la clave privada y el usuario.

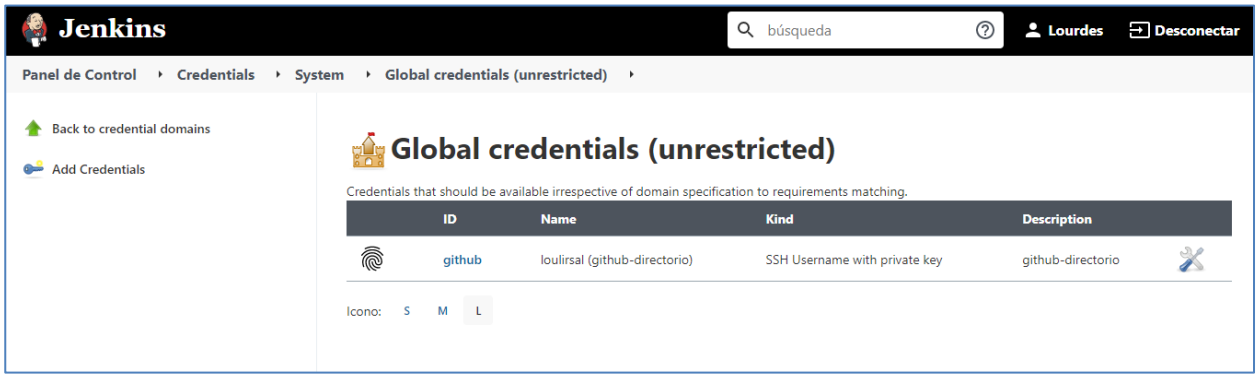

Imagen 29. Credenciales Github en Jenkins

Siguiendo estos pasos, ya se podría descargar el código de GitHub desde Jenkins.

Al haberse instalado Jenkins en local no se puede automatizar la ejecución del Pipeline tras la realización de un push en una rama concreta, pero en el caso en el que se quisiera hacer, simplemente habría que:

- Configurar el webhook en GitHub [105].

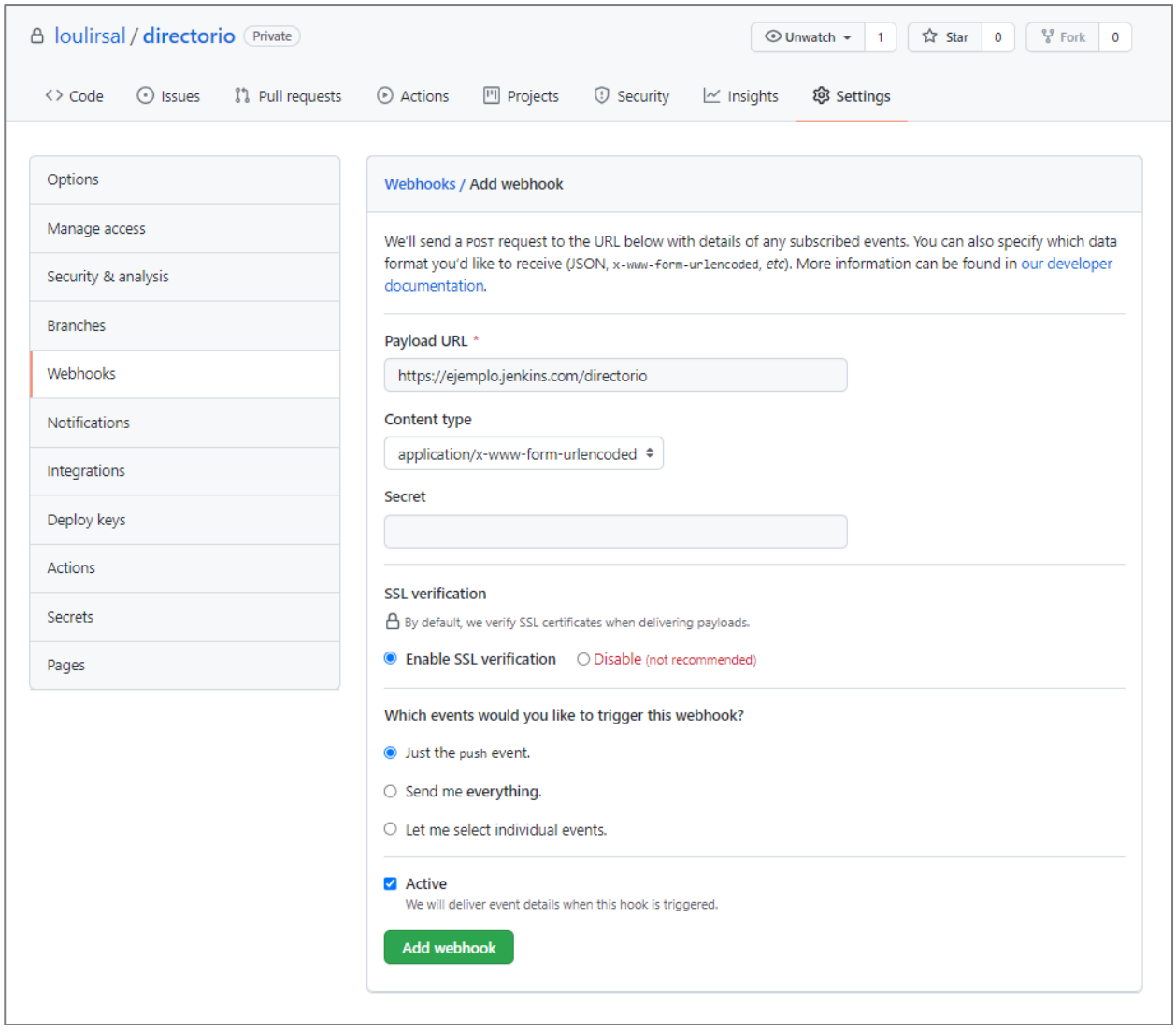

Imagen 30. GitHub: añadir un webhook.

- Configurar la ejecución Jenkins según se indica en la documentación del *plugin* de GitHub [106].

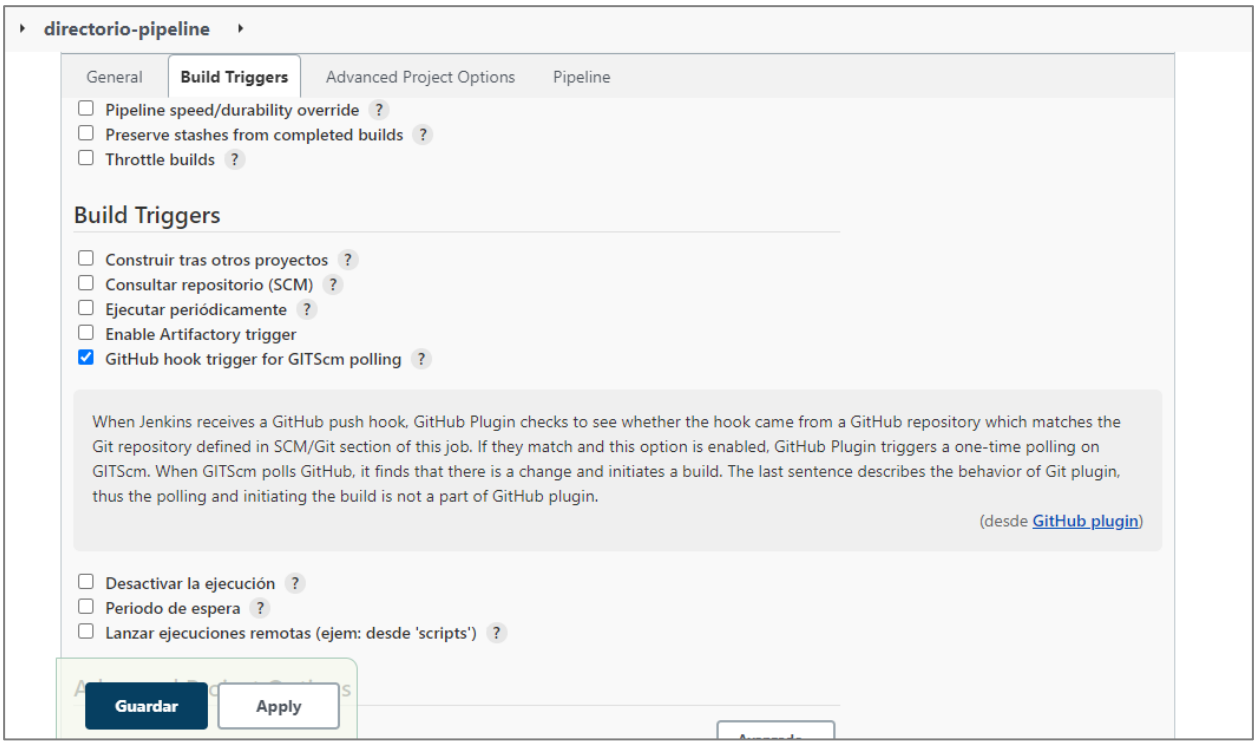

Imagen 31. Jenkins: añadir un trigger para el webhook.

5.10.2 Maven

Para configurar maven habría que hacerlo desde "Administrar Jenkins>Global Tool Configuration":

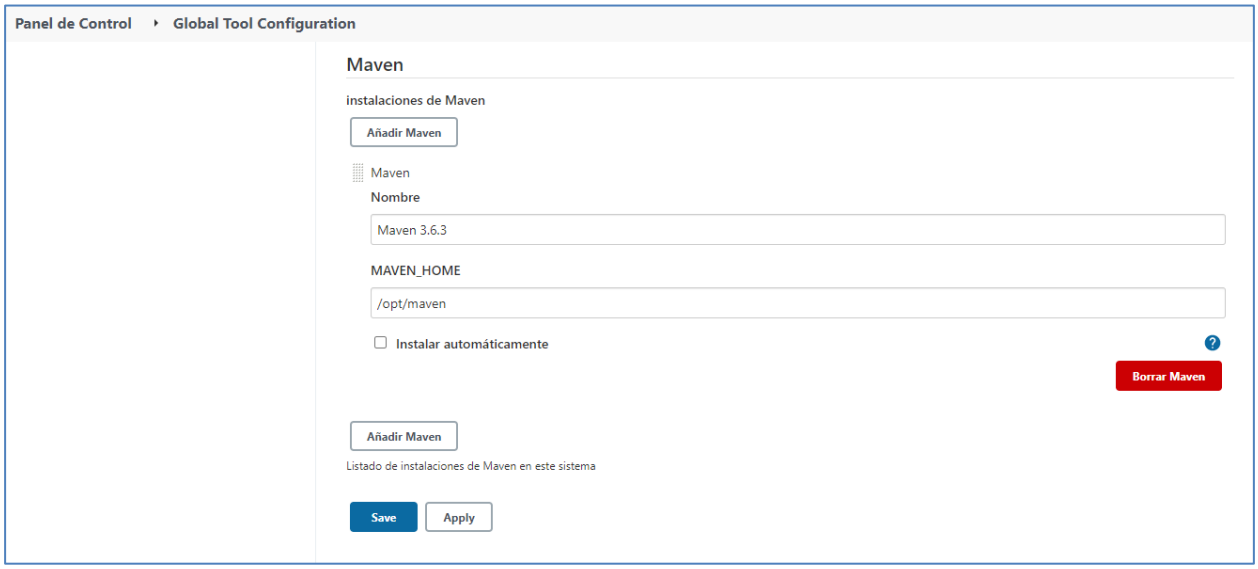

Imagen 32. Configuración Maven en Jenkins

Con esto ya se podría incluir maven en la ejecución de tareas de Jenkins.

5.10.3 Pruebas unitarias

Para integrar la generación de un informe de las pruebas unitarias ejecutadas simplemente habrá que tener instalado en Jenkins el "JUnit Plugin". Cuando se ejecute la cadena de IC, se podrá acceder a los resultados de cada ejecución.

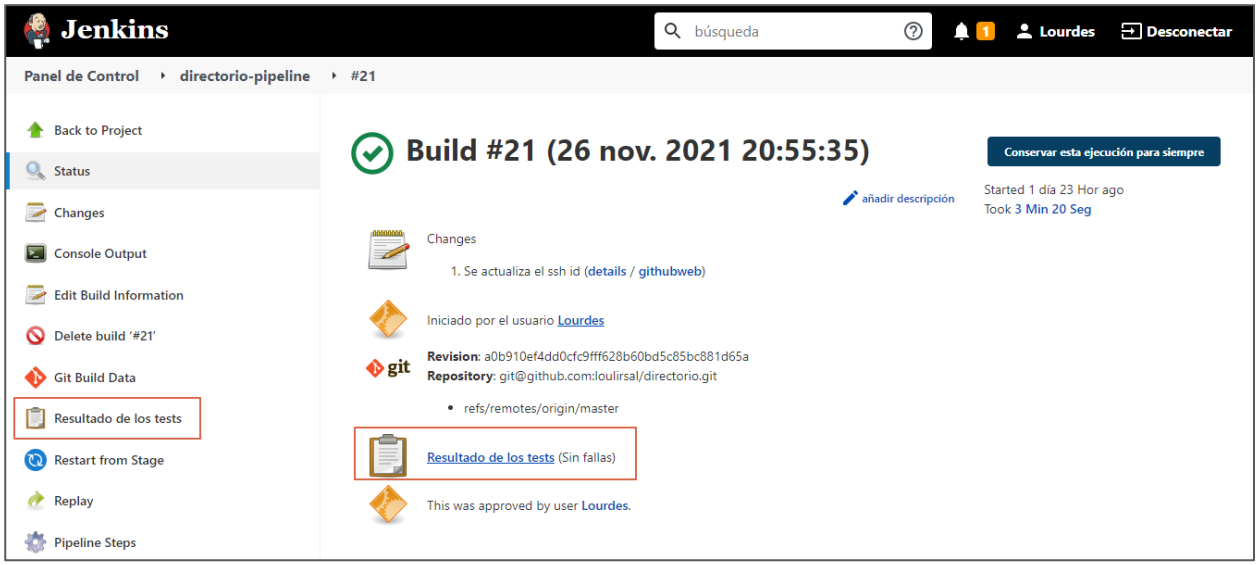

Imagen 33. Acceso a resultados de pruebas unitarias en Jenkins.

También se puede comprobar la tendencia de las pruebas unitarias:

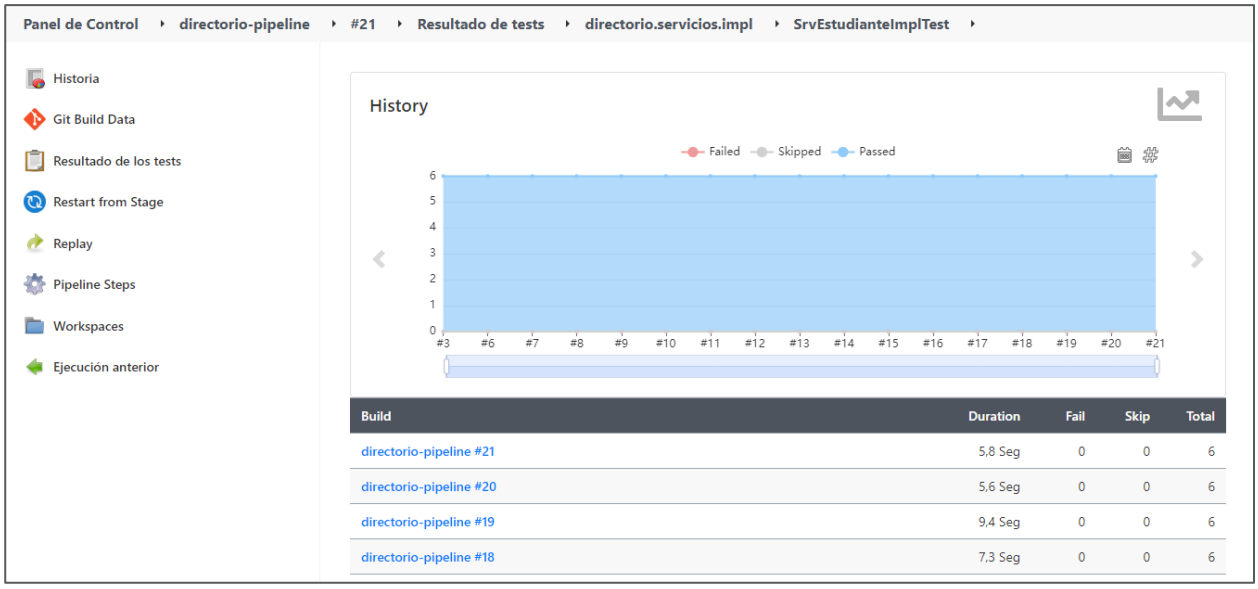

Imagen 34. Tendencia de pruebas unitarias en Jenkins.

Y comprobar el informe de pruebas.

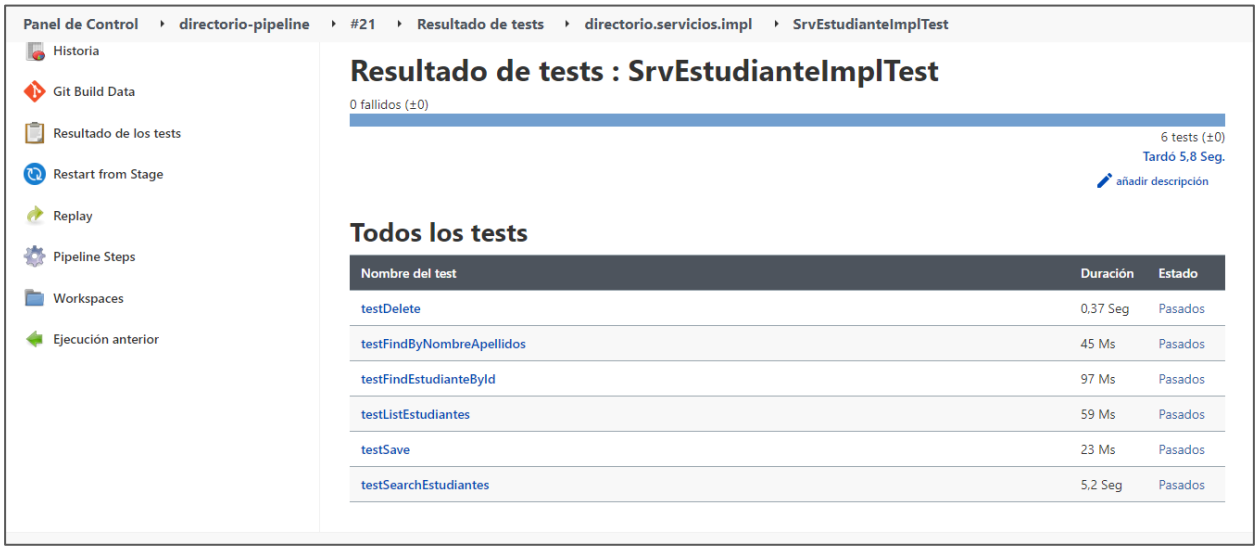

Imagen 35. Informe de pruebas unitarias en Jenkins.

5.10.4 Sonar

En el apartado 5.7 se observa como el mismo Sonar indica cómo integrar Maven con Sonar, por lo que con añadir estas líneas al Jenkinsfile sería suficiente. Una vez ejecutada la cadena de Integración Continua, se podrán observar los resultados obtenidos tras realizar el análisis:

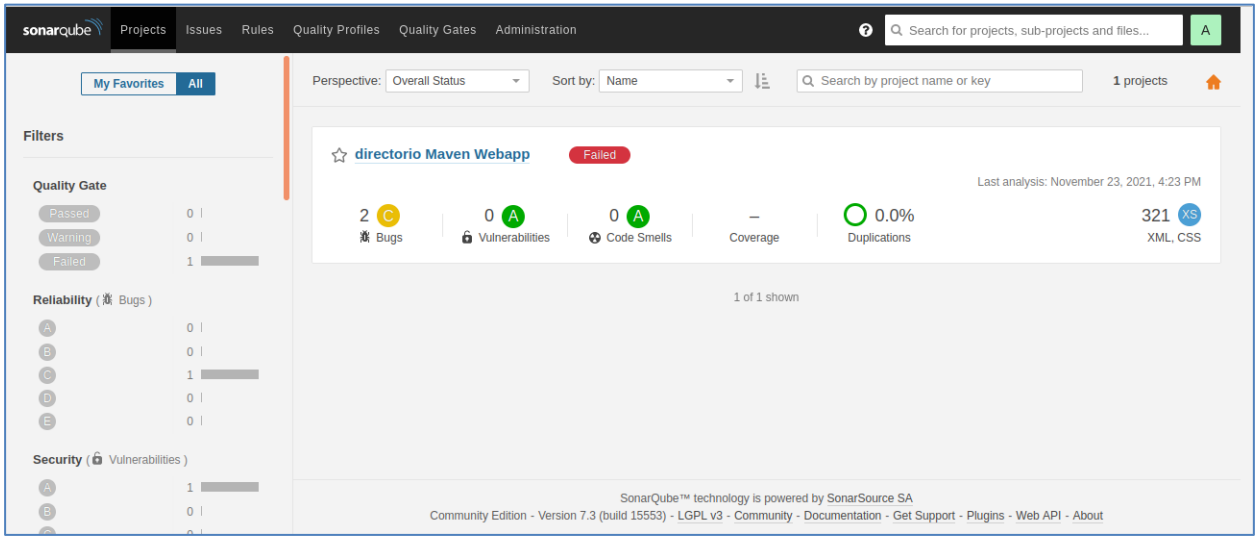

Imagen 36. Primeros resultados en SonarQube.

En este caso se ha obtenido una “C” debido a que se han identificado dos bugs porque se han duplicado líneas en un fichero css del proyecto:

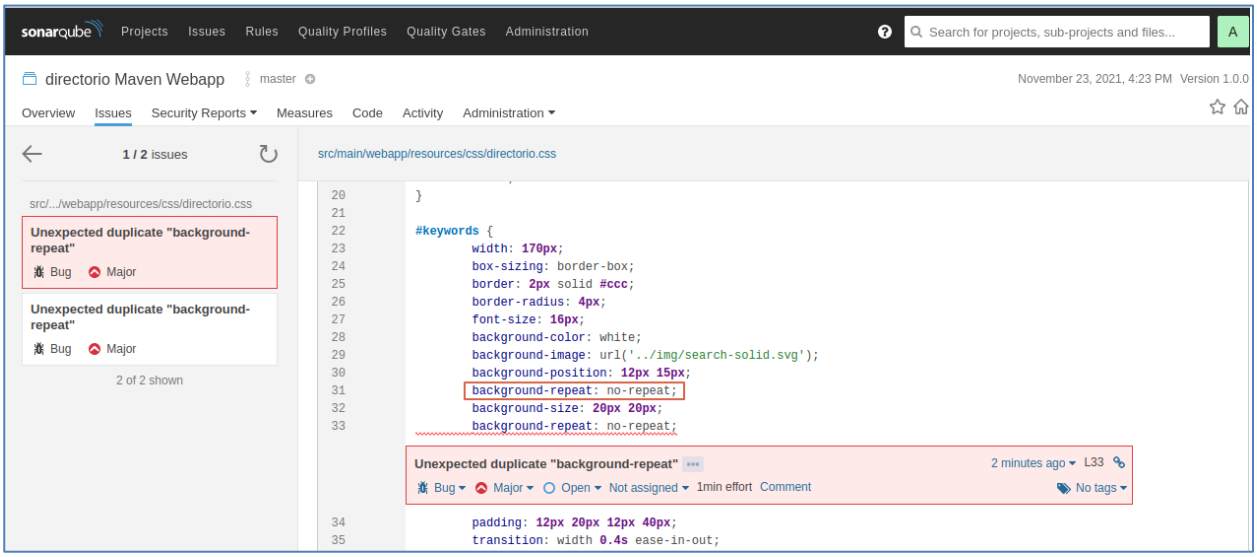

Imagen 37. Bugs en SonarQube.

5.10.5 Artifactory

Para integrar Artifactory se ha seguido la documentación oficial de Artifactory donde se explica cómo desplegar artefactos generados por maven [107].

En primer lugar se ha modificado el pom.xml del proyecto java para incluir la URL del repositorio local donde se van a desplegar los “artefactos”. El código a incluir en el pom.xml se puede obtener desde Artifactory:

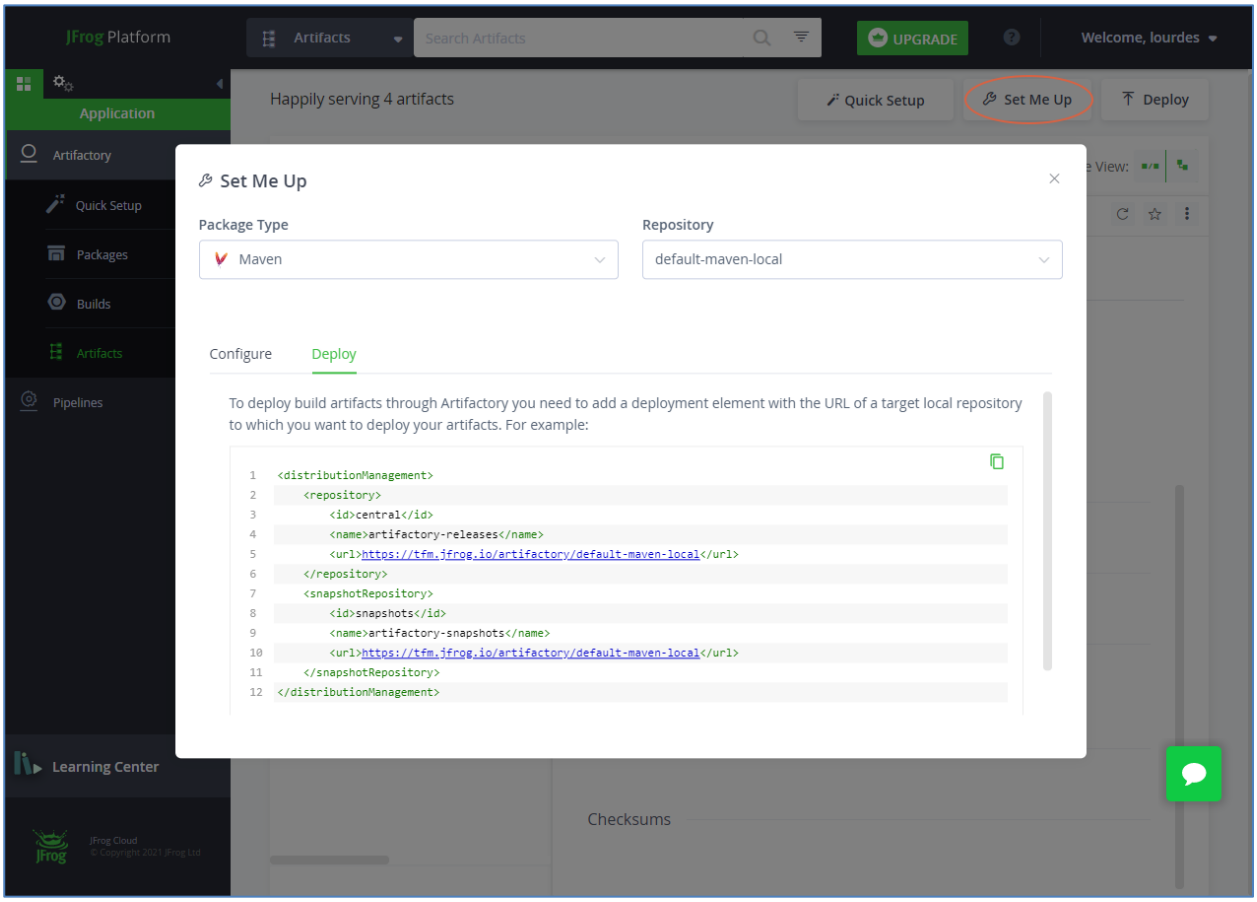

Imagen 38. Configuración pom.xml para desplegar en Artifactory

Tras esto, se ha creado un usuario y se le han dado permisos para el despliegue [108].

Después, se ha incluido un fichero settings.xml en el proyecto java para incluir las credenciales del usuario que se ha creado para desplegar los artefactos.

Una vez realizadas estas configuraciones, bastaría con ejecutar:

```
mvn deploy
```

Con este comando se desplegaría el WAR en Artifactory.

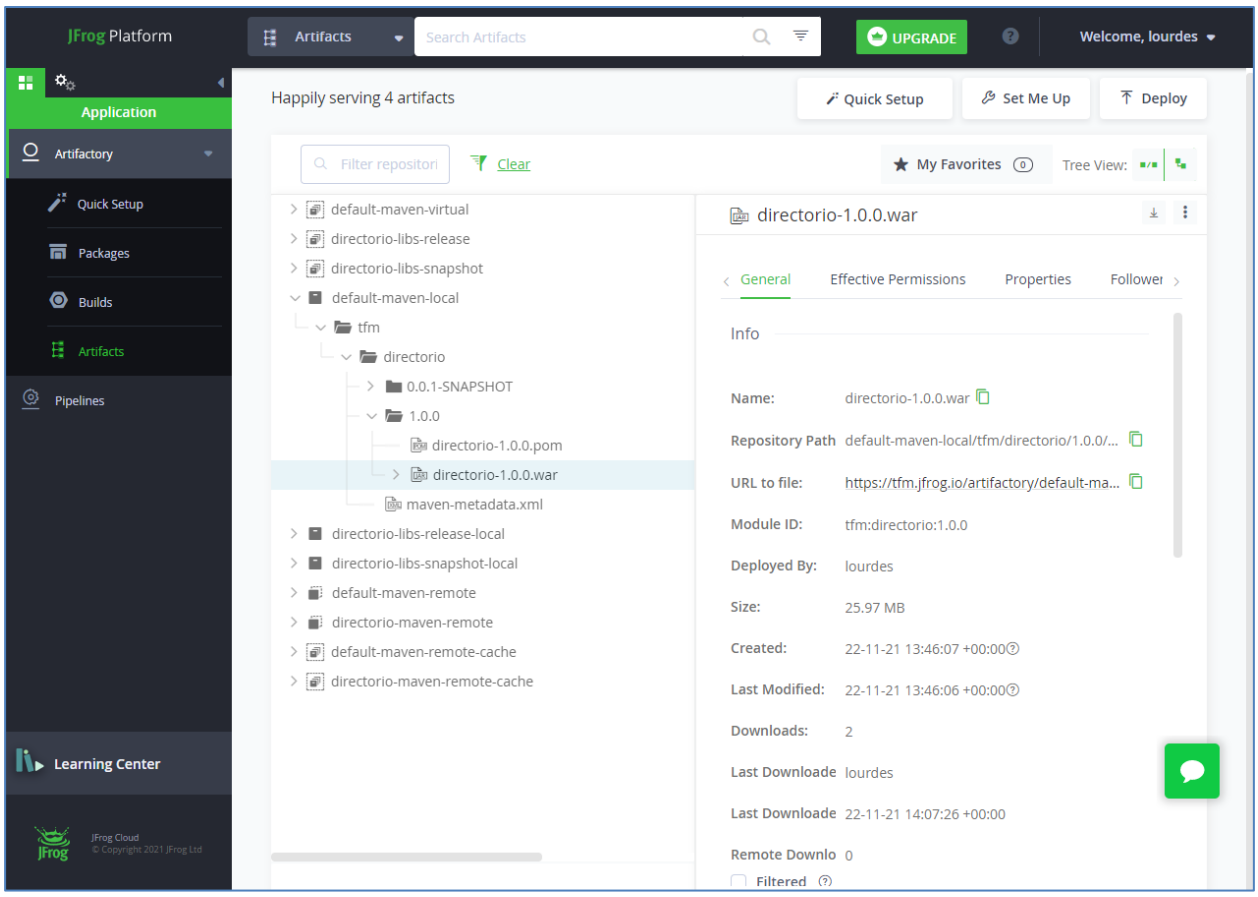

Imagen 39. Artefactos desplegados en Artifactory.

5.10.6 Ansible

Con instalar el *plugin* de Ansible [109] es suficiente para poder ejecutar los Playbooks desde el Pipeline.

Para el despliegue se ha creado un Playbook cuyo código se puede consultar en el anexo 8.4. Este Playbook, incluido en el código fuente, levanta el servidor de aplicaciones Tomcat si no lo estaba ya y despliega la aplicación generada por la cadena de IC de Jenkins.

5.10.7 Creación de pipeline

Para la creación del Pipeline, lo primero es crearlo en el VCS, en este caso incluirlo en el código alojado en GitHub. Así es más fácil gestionarlo que escribiéndolo directamente en una cajetilla de la configuración del Job de Jenkins como ya se ha indicado anteriormente.

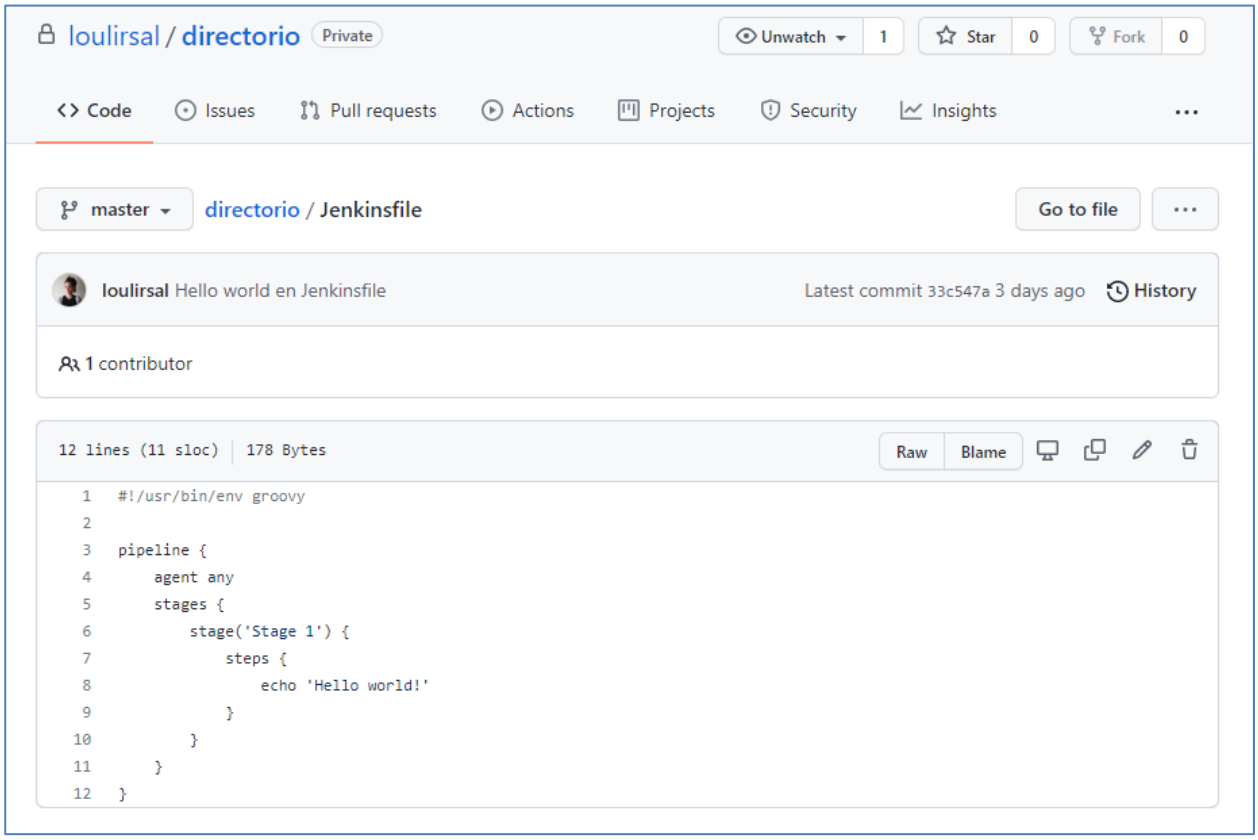

Imagen 40. Jenkinsfile en el SCM

Tras incluirlo, se crea un nuevo Pipeline en Jenkins siguiendo la documentación oficial [110] e indicando de donde se debe descargar, así como los credenciales creados con anterioridad (5.10.1):

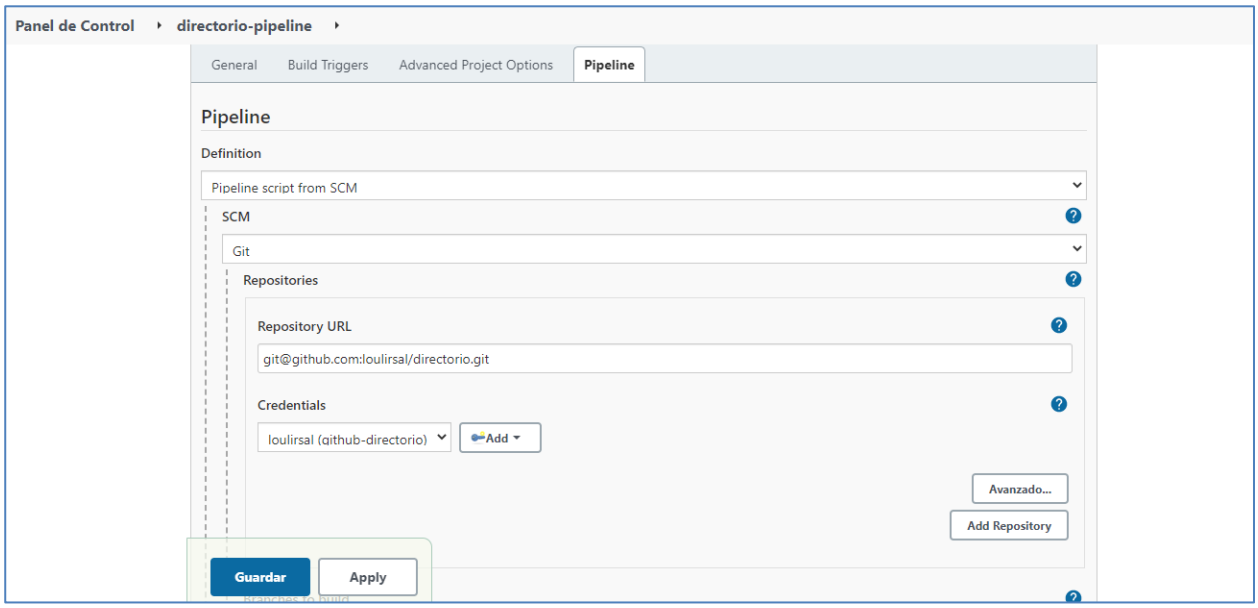

Imagen 41. Acceso a Jenkinsfile desde Pipeline.

Lo primero a configurar sería el *stage* de compilación con maven (ya se explicó su instalación en 5.4 y su integración con Jenkins en 5.10.2). Para ello primero habría que integrar el jdk de la máquina de Jenkins de la

misma forma que se hizo con Maven:

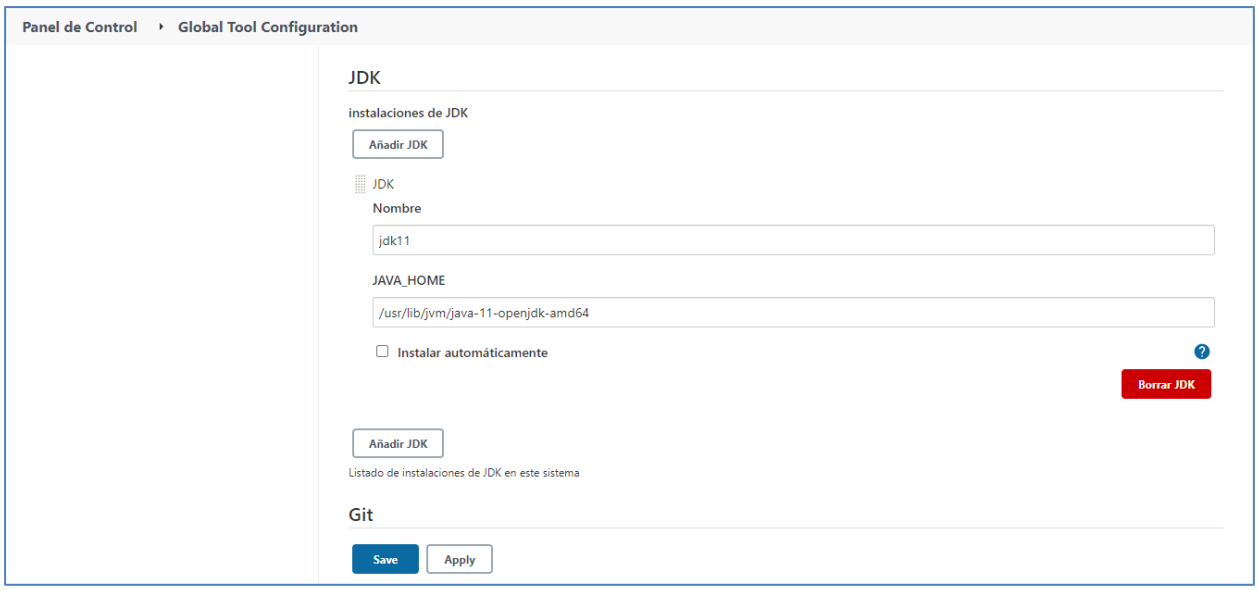

Imagen 42. Integración de JDK con Jenkins.

Luego habría que configurar una sección de herramientas donde incluir las versiones de maven y jdk. También se ha incluido un *stage* de inicialización para mostrar las variables de entorno utilizadas [111].

```
#!/usr/bin/env groovy

pipeline {
    agent any

    tools {
        maven 'Maven 3.6.3'
        jdk 'jdk11'
    }

    stages {

        stage ('Inicialización') {
            steps {
                sh '''
                    echo "PATH = ${PATH}"
                    echo "M2_HOME = ${M2_HOME}"
                '''
            }
        }
```

Imagen 43. Maven y JDK en el Pipeline.

Además, también se incluye a modo de comentario como sería la descarga del código de Git desde el Pipeline. Aunque se haya optado por implementarlo como se indica en la imagen “Acceso a Jenkinsfile desde Pipeline”, esta opción también puede ser útil.

```
/*

Se puede utilizar el siguiente código en caso de que
se quiera realizar la descarga de código fuente del
proyecto sin utilizar el plugin de github:

stage ('Descarga codigo') {
    steps {
        git branch: 'master',
            credentialsId: 'github',
            url: 'git@github.com:loulirsal/directorio.git'
    }
}

*/
```

Imagen 44. Descarga de código de Git en el Pipeline.

Para realizar la compilación bastaría con ejecutar el comando de Maven correspondiente.

```
stage('Compilación - Maven') {
    steps {
        sh 'mvn clean compile'
    }
}
```

Imagen 45.Compilación en el Pipeline.

Para la ejecución de las pruebas unitarias y generación del informe, se ejecuta Maven y luego se utiliza el *plugin* de JUnit para Jenkins para publicar los resultados de las pruebas.

```
stage('Test unitarios - Junit') {
    steps {
        sh 'mvn test'
    }
    post {
        success {
            junit 'target/surefire-reports/**/*.xml'
        }
    }
}
```

Imagen 46. Ejecución de pruebas unitarias y generación de informe en el Pipeline.

Para la ejecución del análisis sonar habría que utilizar el comando Maven:

`sonar:sonar` [112]

Habría que indicarle donde se encuentra la instancia de SonarQube así cómo los datos de la autenticación. Al haberse instalado Jenkins y Sonar en la misma máquina, se indica la dirección local. Para el login se indica el token que ya se generó y que se observa en *Imagen 22. Configuración inicial de SonarQube.*:

```
stage('Análisis estático - Sonar') {
    steps {
        sh 'mvn sonar:sonar -Dsonar.host.url=http://127.0.0.1:9000 -Dsonar.login=754a36a6b6b0e135fc28129cc42560c8650e52e4'
    }
}
```

Imagen 47. Ejecución de análisis en SonarQube desde el Pipeline.

Para la generación del WAR y la subida a Artifactory se utilizan los settings ya descritos tras *Imagen 38. Configuración pom.xml para desplegar en Artifactory*:

```
stage('Generación war y subida - Artifactory') {
    steps {
        sh 'mvn --settings settings.xml deploy -Dmaven.test.skip'
    }
}
```

Imagen 48. Subida de WAR a Artifactory desde el Pipeline.

Tras este último paso se ha incluido un *stage* por si se desea realizar las validaciones manuales pertinentes antes de realizar el despliegue.

```
stage('Aprobación para despliegue') {
    steps {
        input "¿Se aprueba el despliegue?"
    }
}
```

Imagen 49. Aprobación para despliegue en el Pipeline.

Esto podría tener más sentido para la realización de pruebas de aceptación en entornos no productivos antes de la realización del despliegue en entornos productivos.

Esta pregunta se mostraría en la interfaz de usuario de Jenkins y habría que decidir si se continua o se para la cadena de IC

Imagen 50. Logs de aprobación de despliegue en el Pipeline.

Por último se realiza el despliegue con Ansible. Se utiliza el *plugin* AnsiColor [113] para pintar los logs relacionados con Ansible en colores que permitan hacerlos más legibles. Después se utiliza el *plugin* de Ansible para ejecutar el Playbook definido anteriormente.

```
stage('Despliegue con Ansible') {
   steps {

        ansiColor('xterm') {
            ansiblePlaybook(
                playbook: 'playbook-directorio.yml',
                inventory: '/etc/ansible/hosts',
                credentialsId: 'ansible',
                colorized: true)
        }
    }
}
```

Imagen 51. Despliegue con Ansible desde el Pipeline.

Las credenciales a las que se hace referencia son las necesarias para que el usuario que ejecuta Jenkins pueda conectarse mediante SSH a la máquina donde se despliega. Estas credenciales se definen dentro de Jenkins utilizando la funcionalidad “Credentials”:

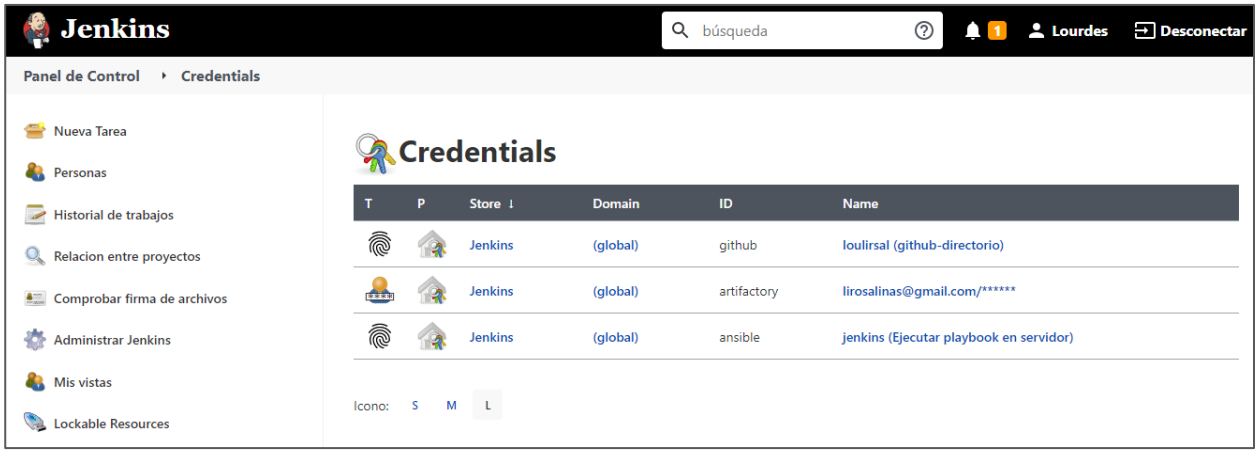

Imagen 52. Credenciales en Jenkins.

6 PRUEBAS REALIZADAS

Ejecución de todo el proceso de IC y DC tras la generación de cambios en el código. Resultados obtenidos.

Una vez lista la creación de todo el proceso de IC y despliegue en la arquitectura creada para ello, se han realizado varias ejecuciones de la cadena para realizar pruebas representativas de su funcionamiento.

Para poner en contexto al lector se muestra como se observan en Jenkins las ejecuciones de los Pipelines:

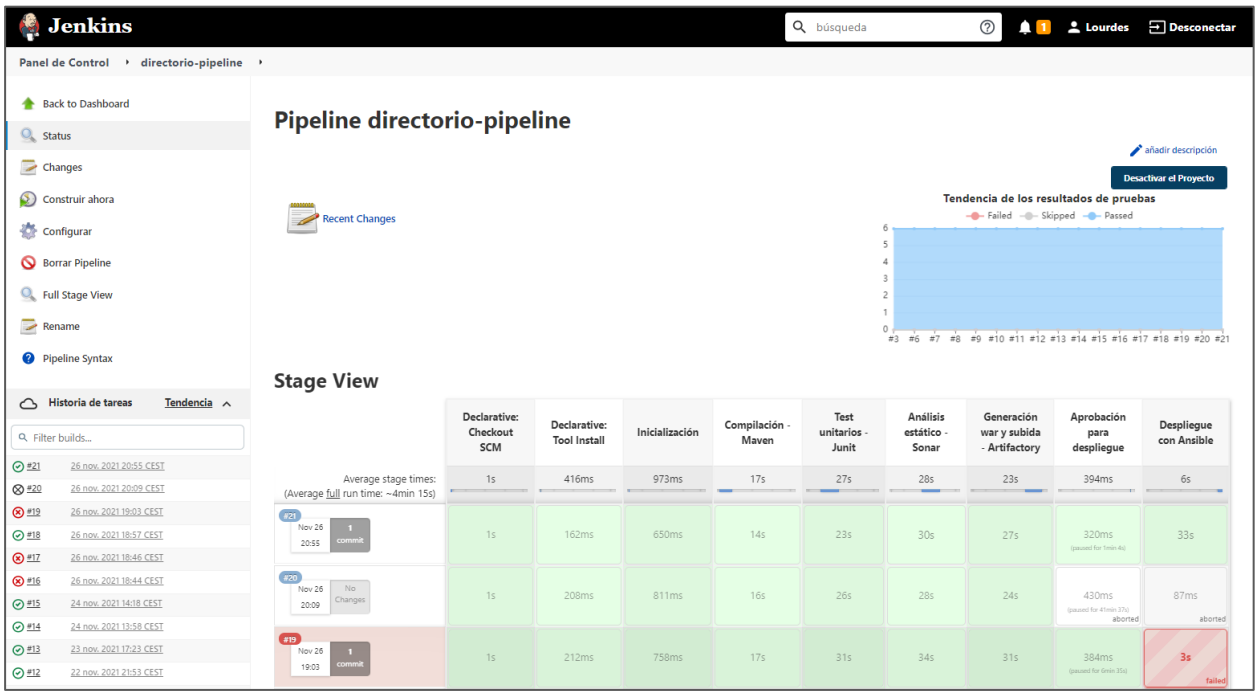

Imagen 53. Ejecuciones.

En la imagen se observa en distintos colores el resultado de la ejecución de los distintos pasos:

- En verde cuando se ejecuta con éxito. Se observa que la ejecución #21 se ha ejecutado correctamente en su totalidad. El resultado de la ejecución es “SUCCESS”.
- En gris cuando no se ejecuta. Se observa en la ejecución #20, donde se había indicado que no se aprobaba el despliegue. El resultado de la ejecución es “ABORTED”.
- En rojo cuando se producen errores. En el caso de la ejecución #19 se produjo porque no se habían indicado correctamente las credenciales utilizadas por Ansible para la conexión SSH. El resultado de la ejecución es “FAILURE”.

Jenkins también permite obtener la tendencia del tiempo de ejecución del Pipeline.

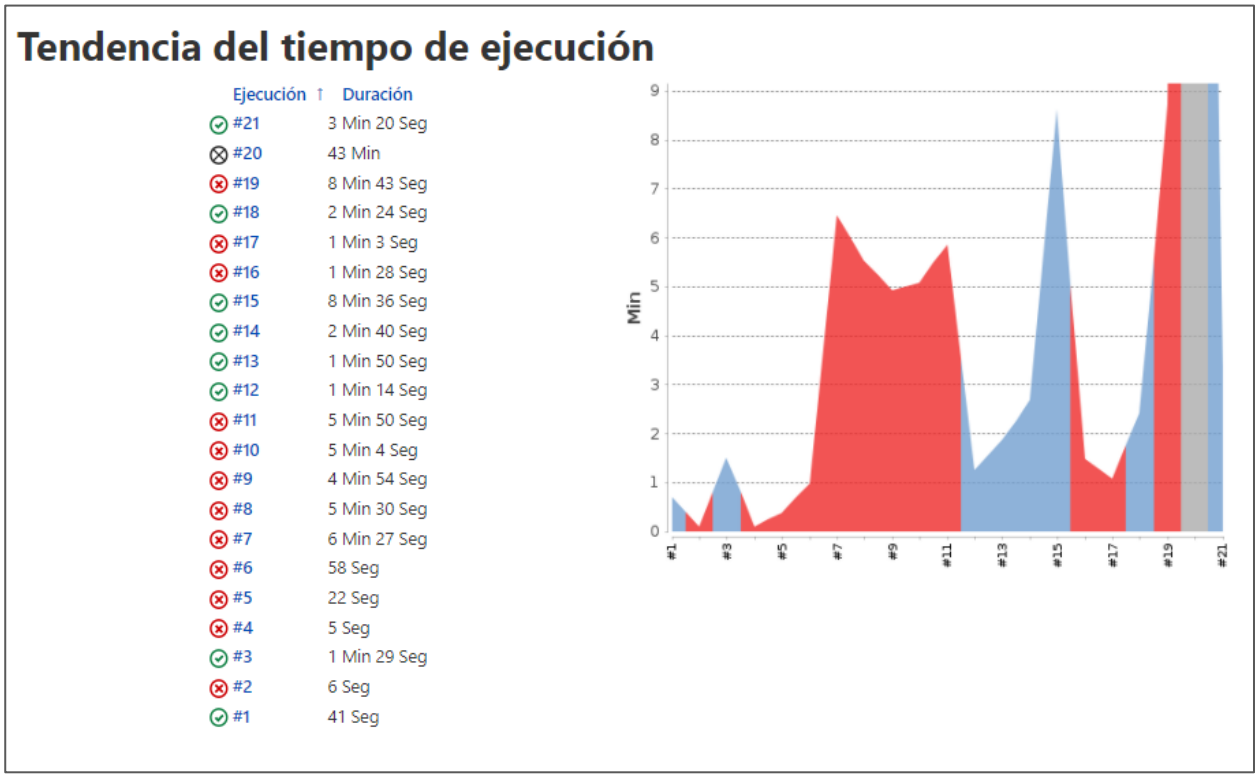

Imagen 54. Tendencia tiempo de ejecución.

Una vez visto esto, se procede a describir distintas pruebas:

6.1 Instalación de Maven y JDK

Si no se indica en la configuración de Jenkins las instalaciones de Maven y JDK a utilizar, se produce un error del tipo que se aprecia en la siguiente imagen.

Imagen 55. Pruebas: instalación de Maven y JDK.

Se soluciona indicando las instalaciones de Maven y JDK en el equipo utilizando la configuración de sistema de Jenkins.

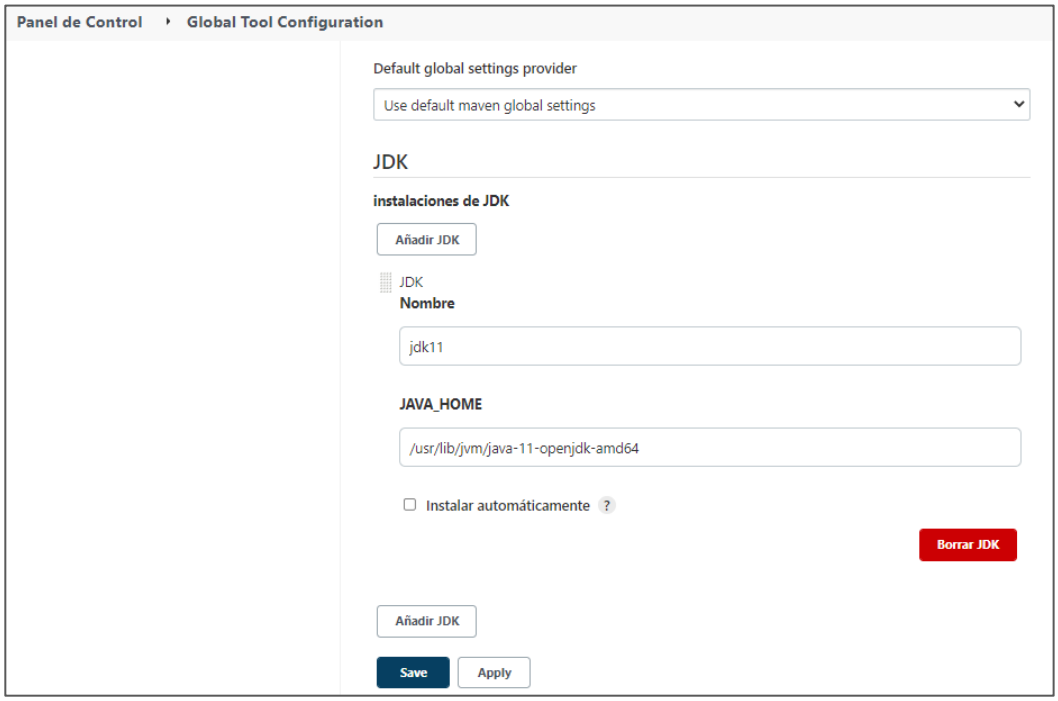

Imagen 56. Pruebas: instalación de JDK.

6.2 Error de sintaxis

Cuando se produce algún error en la sintaxis del Pipeline, también se produce un error en la ejecución:

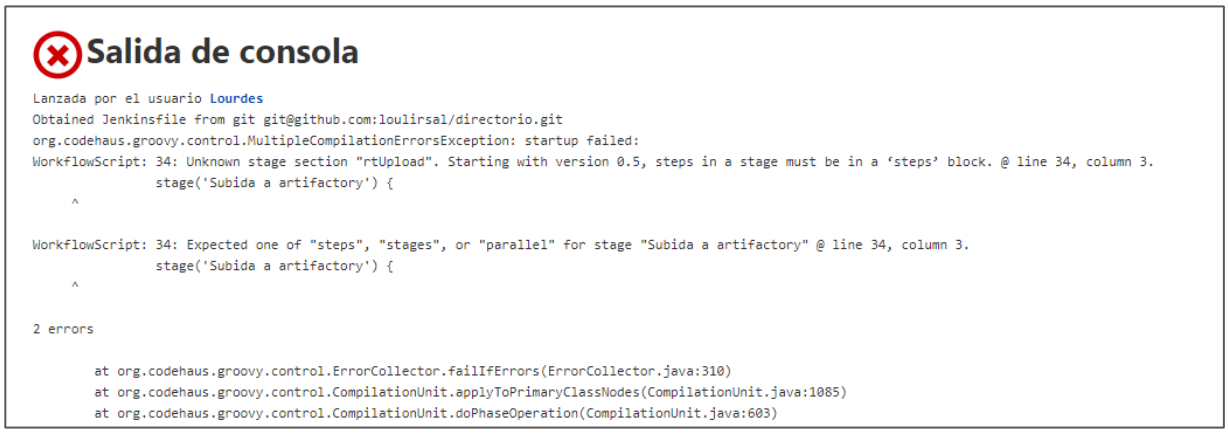

Imagen 57. Pruebas: error de sintaxis.

Para evitar esto, existe una herramienta que ofrece Jenkins y que permite comprobar la sintaxis de cualquier directiva que se quiera generar:

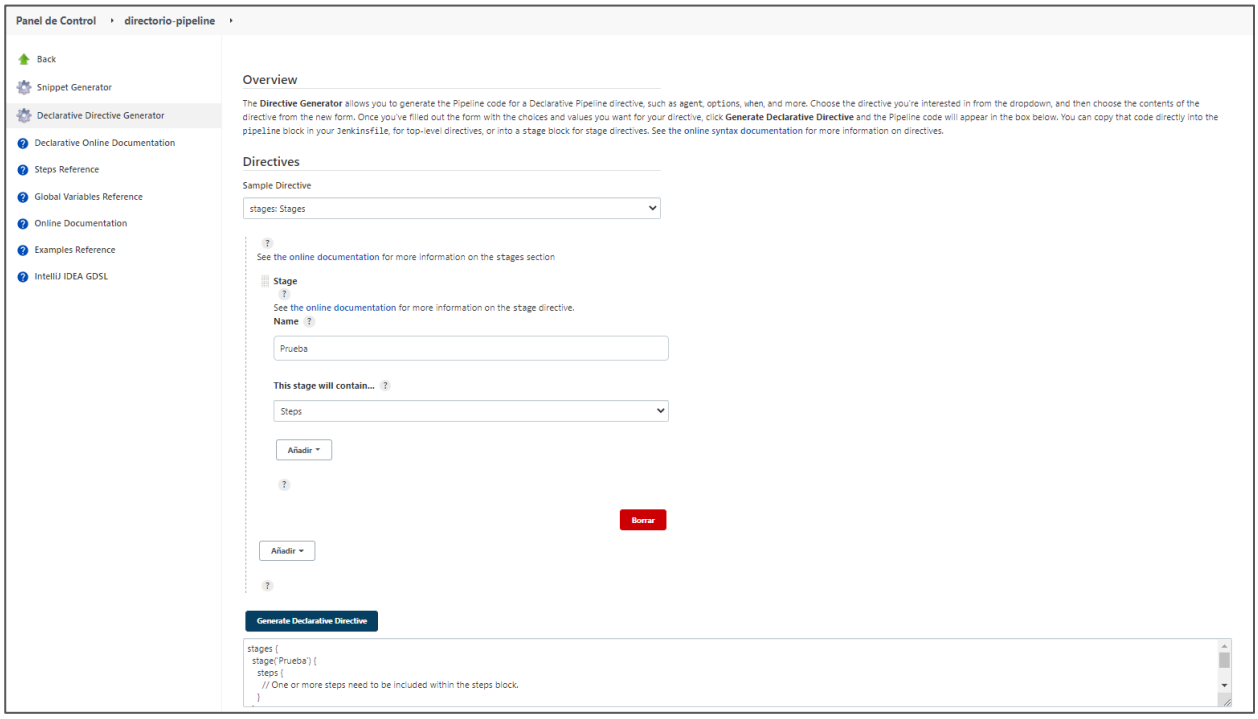

Imagen 58. Pruebas: funcionamiento de Pipeline Syntax.

Esta herramienta es "Pipeline Syntax" accesible desde la configuración de la tarea de Jenkins:

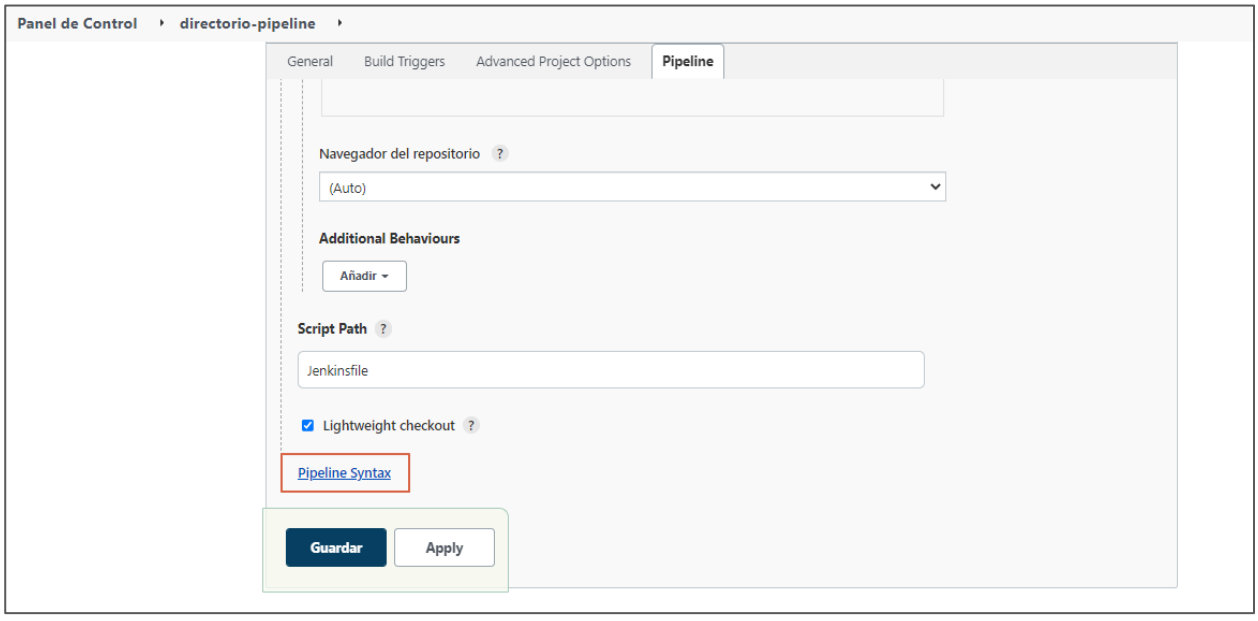

Imagen 59. Pruebas: acceso a Pipeline Syntax.

6.3 No se continúa ante un error

En esta prueba, SonarQube no estaba funcionando correctamente, por lo que falla el análisis estático de código y por lo tanto no se continúa con el resto de pasos:

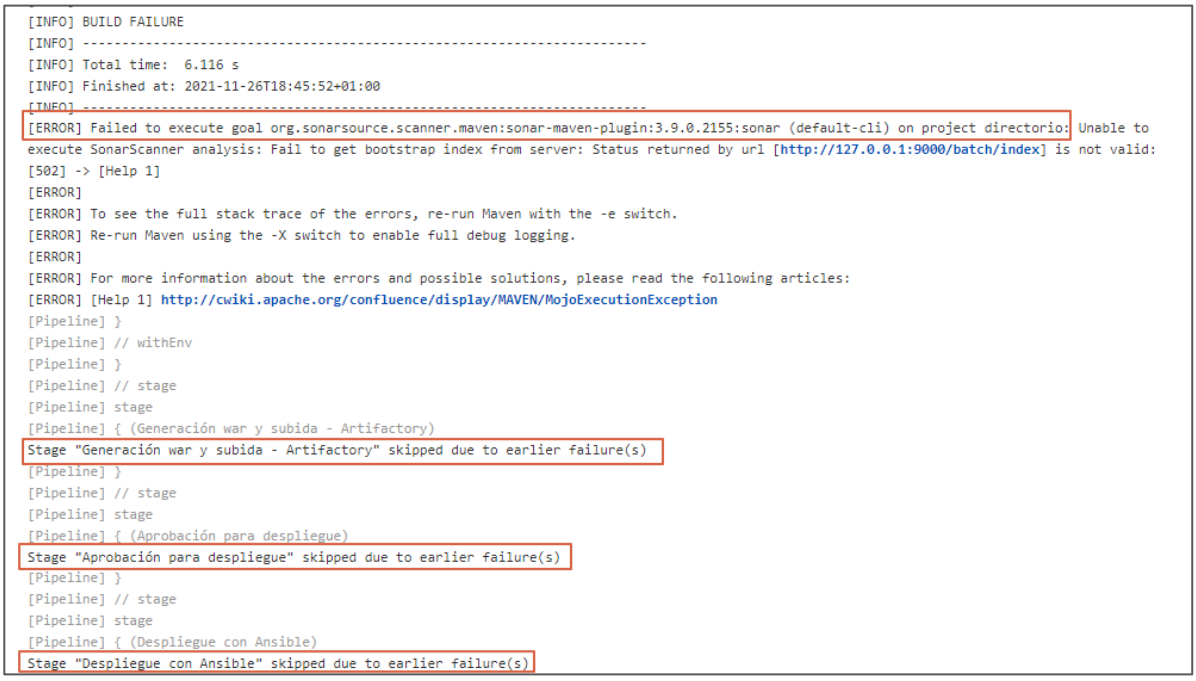

Imagen 60. Pruebas: no se continúa ante un error.

En este caso se considera un comportamiento correcto, pero podría permitirse que se continuara aunque fallara mediante el uso de la directiva “catchError” [114]

6.4 No se acepta el despliegue

Si no se acepta el despliegue en el paso en el que se consulta, el Pipeline acaba con estado “ABORTED” en lugar de “FAILURE” y no se continúa con los siguientes pasos:

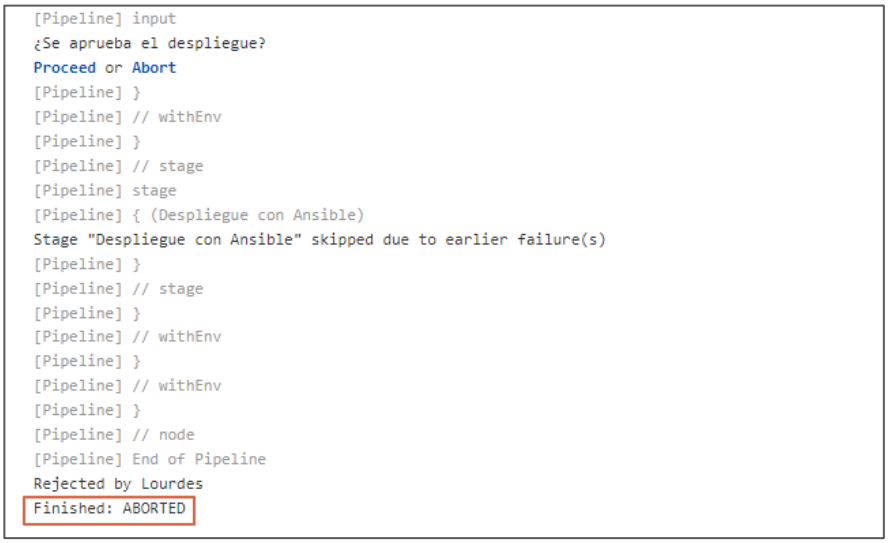

Imagen 61. Pruebas: no se acepta el despliegue.

6.5 Ejecución realizada con éxito

Por último se ha realizado una ejecución completa con éxito donde se puede observar que se termina desplegando la aplicación "directorio" en el servidor Tomcat y que la salida de la ejecución es "SUCCESS".

```
TASK [Levantar tomcat si no lo esta] ****************************************
ok: [34.125.204.166]

TASK [Desplegar directorio en tomcat] ***************************************
changed: [34.125.204.166]

PLAY RECAP ******************************************************************
34.125.204.166             : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[Pipeline] }

[Pipeline] // ansiColor
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Imagen 62. Pruebas: ejecución realizada con éxito.

7 CONCLUSIONES

Para obtener las conclusiones se realiza una valoración de resultados obtenidos haciendo referencia al estudio realizado, se identifican puntos de mejora y se indica sobre qué sería interesante profundizar.

En primer lugar resulta interesante partir de lo que proponía Martin Fowler en la lectura referenciada al principio de este documento [1]. El autor definía la Integración Continua haciendo referencia a una serie de principios que se han abarcado en este proyecto.

Habla de mantener un único repositorio de código fuente accesible para todos los desarrolladores, lo cual se ha implementado utilizando git y GitHub; se ha automatizado la compilación utilizando Maven y por supuesto Jenkins y se han creado pruebas unitarias, un plan de pruebas funcional y un análisis de código estático, cubriendo también el requisito de la realización de pruebas.

En cuanto a la realización de una compilación con los nuevos cambios todos los días en la máquina de integración, se hace posible gracias a la utilización de Jenkins y GitHub, donde aparecen dos opciones para cumplir con esta recomendación:

- Automatización de ejecución diaria del Pipeline de Jenkins mediante el "Build Trigger" de ejecución periódica.
- Activación del webhook de GitHub según se ha indicado en 5.10.1.

Se consigue que sea fácil no solo obtener el último ejecutable generado, sino cada uno de ellos al almacenarlos en Artifactory. También se automatiza el despliegue gracias al uso de Jenkins y Ansible, lo cual permiten realizar un acercamiento al concepto de Despliegue Continuo que también se ha mencionado.

Todo esto utilizando una herramienta como Jenkins, que permite que cualquier participante del proyecto pueda seguir el estado actual del mismo simplemente accediendo a esta herramienta.

Por otra parte se trabaja el concepto DevOps al unificar procesos propios de los ingenieros de desarrollo, aseguramiento de calidad y sistemas en un proceso automatizado en el que se incorporan fases de cada uno de los campos.

También se está facilitando el funcionamiento de proyectos Agile, ya que se permite que sea muy fácil para el cliente y usuario final observar los avances y mantener un diálogo más fluido con el equipo de desarrollo. Permite que los desarrolladores pueden observar el estado de la aplicación en un entorno de integración y realizar las pruebas funcionales correspondientes en un entorno con la que podría ser la versión liberada en una futura entrega sin tener que esperar a una integración final, donde el riesgo sería mucho mayor. También facilita mucho el seguimiento de proyectos Scrum, donde la metodología implica que se realicen demostraciones continuas a los stakeholders [115] del proyecto.

En cuanto a los puntos de mejora se proponen varios:

- Realización de pruebas de regresión automatizadas con herramientas como Selenium [116], Katalon [117], RobotFramework [118] o Cypress [119].
- Gestión de marcha atrás ante fallos en pruebas de regresión automatizadas, lo cual se podría realizar con ShellScript y/o Ansible.
- Automatización de pruebas de rendimiento, con herramientas como Apache JMeter [120].
- Incorporación de herramientas para la gestión de cambios en base de datos como Liquibase [121] o DBmaestro [122].

Cómo siguientes conceptos en los que se podría profundizar a partir de este proyecto sería interesante continuar creando casos de prueba para despliegues continuos en distintos entornos así como incorporar la utilización de Dockers que es una tecnología que se utiliza en cada vez más proyectos en la actualidad.

8 ANEXOS

8.1 Scripts de BBDD

8.1.1 00_BaseDatos.sql

```
--
--  00_BaseDatos.sql
--  Creación de la base de datos
--

CREATE DATABASE directorio;
```

8.1.2 01_Estudiante.sql

```
--
--  01_Estudiante.sql
--  Creación de la tabla Estudiante
--

CREATE TABLE estudiante (
  id INT NOT NULL AUTO_INCREMENT COMMENT 'Identificador del estudiante',
  nombre VARCHAR(45) NOT NULL COMMENT 'Nombre del estudiante',
  apellidos VARCHAR(128) NOT NULL COMMENT 'Apellidos del estudiante',
  nacimiento DATETIME NOT NULL COMMENT 'Fecha de nacimiento del estudiante',
  correo VARCHAR(45) NOT NULL COMMENT 'Correo electrónico del estudiante',
  PRIMARY KEY (id))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8
COLLATE = utf8_spanish_ci
COMMENT = 'Tabla donde se almacenan los datos personales de los estudiantes';
```

8.1.3 00_Estudiante_rollback.sql

```
--
--  00_Estudiante_rollback.sql
--  Borrado de la tabla Estudiante
--

DROP TABLE estudiante;
```

8.1.4 01_ BaseDatos _rollback.sql

```
--
--  01_BaseDatos_rollback.sql
--  Borrado de la base de datos
--

DROP DATABASE directorio;
```

8.2 Jenkinsfile

```
pipeline {
    agent any

      tools {
       maven 'Maven 3.6.3'
       jdk 'jdk11'
    }

    stages {

      stage ('Inicialización') {
          steps {
               sh '''
                   echo "PATH = ${PATH}"
                   echo "M2_HOME = ${M2_HOME}"
               '''
          }
       }

       /*

       Se puede utilizar el siguiente código en caso de que
       se quiera realizar la descarga de código fuente del
       proyecto sin utilizar el plugin de github:

       stage ('Descarga codigo') {
           steps {
               git branch: 'master',
                   credentialsId: 'github',
                   url: 'git@github.com:loulirsal/directorio.git'
           }
       }

       */

       stage('Compilación - Maven') {
           steps {
               sh 'mvn clean compile'
           }
       }

       stage('Test unitarios - Junit') {
           steps {
               sh 'mvn test'
           }
           post {
               success {
                   junit 'target/surefire-reports/**/*.xml'
               }
           }
       }

       stage('Análisis estático - Sonar') {
           steps {
               sh 'mvn sonar:sonar -Dsonar.host.url=http://127.0.0.1:9000 -
Dsonar.login=754a36a6b6b0e135fc28129cc42560c8650e52e4'
```

```
            }
        }

        stage('Generación war y subida - Artifactory') {
            steps {
                sh 'mvn --settings settings.xml deploy -Dmaven.test.skip'
            }
        }

        stage('Aprobación para despliegue') {
            steps {
                input "¿Se aprueba el despliegue?"
            }
        }

         stage('Despliegue con Ansible') {
            steps {
                ansiColor('xterm') {
                    ansiblePlaybook(
                    playbook: 'playbook-directorio.yml',
                    inventory: '/etc/ansible/hosts',
                    credentialsId: 'ansible',
                    colorized: true)
                }
            }
         }
    }
}
```

8.3 Informe pruebas

8.4 Playbook de Ansible

```
---
- hosts: servidor
  become: yes
  tasks:
  - name: Levantar tomcat si no lo esta
    service: name=tomcat state=started enabled=yes
  - name: Desplegar directorio en tomcat
    ansible.builtin.copy:
      src: target/directorio.war
      dest: /opt/tomcat/webapps/


...
```

8.5 Salida de consola de Jenkins

```
Lanzada por el usuario Lourdes
Obtained Jenkinsfile from git git@github.com:loulirsal/directorio.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins  in /var/lib/jenkins/workspace/directorio-pipeline
[Pipeline] {
[Pipeline] stage
```

```
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
using credential github
 > git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/directorio-
pipeline/.git # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url git@github.com:loulirsal/directorio.git #
timeout=10
Fetching upstream changes from git@github.com:loulirsal/directorio.git
 > git --version # timeout=10
 > git --version # 'git version 2.17.1'
using GIT_SSH to set credentials github-directorio
 > git fetch --tags --progress -- git@github.com:loulirsal/directorio.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision a0b910ef4dd0cfc9fff628b60bd5c85bc881d65a
(refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f a0b910ef4dd0cfc9fff628b60bd5c85bc881d65a # timeout=10
Commit message: "Se actualiza el ssh id"
 > git rev-list --no-walk f98b8fee2b7ecb326a21a9d590f9b7b97abaf54f #
timeout=10
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Tool Install)
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Inicialización)
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] withEnv
[Pipeline] {
[Pipeline] sh
+ echo PATH = /usr/lib/jvm/java-11-openjdk-
amd64/bin:/opt/maven/bin:/usr/lib/jvm/java-11-openjdk-
amd64/bin:/opt/maven/bin:/opt/maven/bin:/usr/local/sbin:/usr/local/bin:/usr/s
bin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
PATH = /usr/lib/jvm/java-11-openjdk-
amd64/bin:/opt/maven/bin:/usr/lib/jvm/java-11-openjdk-
amd64/bin:/opt/maven/bin:/opt/maven/bin:/usr/local/sbin:/usr/local/bin:/usr/s
bin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
+ echo M2_HOME = /opt/maven
M2_HOME = /opt/maven
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Compilación - Maven)
[Pipeline] tool
```

```
[Pipeline] envVarsForTool
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] withEnv
[Pipeline] {
[Pipeline] sh
+ mvn clean compile
[INFO] Scanning for projects...
[INFO]
[INFO] ---------------------------< tfm:directorio >-------------------------
--
[INFO] Building directorio Maven Webapp 1.0.0
[INFO] --------------------------------[ war ]---------------------------------
--
[INFO]
[INFO] --- maven-clean-plugin:3.1.0:clean (default-clean) @ directorio ---
[INFO] Deleting /var/lib/jenkins/workspace/directorio-pipeline/target
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:resources (default-resources) @
directorio ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 2 resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:compile (default-compile) @ directorio
---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 8 source files to /var/lib/jenkins/workspace/directorio-
pipeline/target/classes
[INFO] ------------------------------------------------------------------------
--
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
--
[INFO] Total time:  8.688 s
[INFO] Finished at: 2021-11-26T20:55:54+01:00
[INFO] ------------------------------------------------------------------------
--
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Test unitarios - Junit)
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] withEnv
[Pipeline] {
[Pipeline] sh
+ mvn test
[INFO] Scanning for projects...
[INFO]
[INFO] ---------------------------< tfm:directorio >-------------------------
--
[INFO] Building directorio Maven Webapp 1.0.0
[INFO] --------------------------------[ war ]---------------------------------
--
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:resources (default-resources) @
directorio ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
```

```
[INFO] Copying 2 resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:compile (default-compile) @ directorio
---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:testResources (default-testResources)
@ directorio ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory
/var/lib/jenkins/workspace/directorio-pipeline/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:testCompile (default-testCompile) @
directorio ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to /var/lib/jenkins/workspace/directorio-
pipeline/target/test-classes
[INFO]
[INFO] --- maven-surefire-plugin:2.22.2:test (default-test) @ directorio ---
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running directorio.servicios.impl.SrvEstudianteImplTest
[INFO] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 6.009
s - in directorio.servicios.impl.SrvEstudianteImplTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
--
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
--
[INFO] Total time:  18.407 s
[INFO] Finished at: 2021-11-26T20:56:18+01:00
[INFO] ------------------------------------------------------------------------
--
Post stage
[Pipeline] junit
Grabando resultados de tests
[Checks API] No suitable checks publisher found.
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Análisis estático - Sonar)
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] withEnv
[Pipeline] {
[Pipeline] sh
+ mvn sonar:sonar -Dsonar.host.url=http://127.0.0.1:9000 -
Dsonar.login=754a36a6b6b0e135fc28129cc42560c8650e52e4
[INFO] Scanning for projects...
[INFO]
```

```
[INFO] ---------------------------< tfm:directorio >------------------------
--
[INFO] Building directorio Maven Webapp 1.0.0
[INFO] --------------------------------[ war ]---------------------------------
--
[INFO]
[INFO] --- sonar-maven-plugin:3.9.0.2155:sonar (default-cli) @ directorio ---
[INFO] User cache: /var/lib/jenkins/.sonar/cache
[INFO] SonarQube version: 7.3.0
[INFO] Default locale: "en_US", source code encoding: "UTF-8"
[INFO] Publish mode
[INFO] Load global settings
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by com.google.protobuf.UnsafeUtil
(file:/var/lib/jenkins/.sonar/cache/da9310bb9fdabea7c7071706af6a955b/sonar-
scanner-engine-shaded-7.3-all.jar) to field java.nio.Buffer.address
WARNING: Please consider reporting this to the maintainers of
com.google.protobuf.UnsafeUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal
reflective access operations
WARNING: All illegal access operations will be denied in a future release
[INFO] Load global settings (done) | time=363ms
[INFO] Server id: A7EE8CF2-AX1M3MwQa2dhfyHuGy0H
[INFO] User cache: /var/lib/jenkins/.sonar/cache
[INFO] Load/download plugins
[INFO] Load plugins index
[INFO] Load plugins index (done) | time=179ms
[INFO] Load/download plugins (done) | time=336ms
[INFO] Loaded core extensions:
[INFO] Process project properties
[INFO] Load project repositories
[INFO] Load project repositories (done) | time=461ms
[INFO] Load quality profiles
[INFO] Load quality profiles (done) | time=196ms
[INFO] Load active rules
[INFO] Load active rules (done) | time=1828ms
[INFO] Load metrics repository
[INFO] Load metrics repository (done) | time=156ms
[INFO] Project key: tfm:directorio
[INFO] Project base dir: /var/lib/jenkins/workspace/directorio-pipeline
[INFO] -------------  Scan directorio Maven Webapp
[INFO] Base dir: /var/lib/jenkins/workspace/directorio-pipeline
[INFO] Working dir: /var/lib/jenkins/workspace/directorio-
pipeline/target/sonar
[INFO] Source paths: src/main/webapp, pom.xml, src/main/java
[INFO] Test paths: src/test/java
[INFO] Source encoding: UTF-8, default locale: en_US
[INFO] Load server rules
[INFO] Load server rules (done) | time=673ms
[INFO] Index files
[INFO] 17 files indexed
[INFO] Quality profile for css: Sonar way
[INFO] Quality profile for xml: Sonar way
[INFO] Sensor SonarCSS Metrics [cssfamily]
[WARNING] Metric 'comment_lines_data' is deprecated. Provided value is
ignored.
[INFO] Sensor SonarCSS Metrics [cssfamily] (done) | time=270ms
[INFO] Sensor SonarCSS Rules [cssfamily]
[INFO] Sensor SonarCSS Rules [cssfamily] (done) | time=4087ms
[INFO] Sensor XML Sensor [xml]
[INFO] Sensor XML Sensor [xml] (done) | time=995ms
[INFO] Sensor Zero Coverage Sensor
```

```
[INFO] Sensor Zero Coverage Sensor (done) | time=37ms
[INFO] Calculating CPD for 0 files
[INFO] CPD calculation finished
[INFO] Analysis report generated in 588ms, dir size=46 KB
[INFO] Analysis reports compressed in 88ms, zip size=17 KB
[INFO] Analysis report uploaded in 71ms
[INFO] ANALYSIS SUCCESSFUL, you can browse
http://127.0.0.1:9000/dashboard?id=tfm%3Adirectorio
[INFO] Note that you will be able to access the updated dashboard once the
server has processed the submitted analysis report
[INFO] More about the report processing at
http://127.0.0.1:9000/api/ce/task?id=AX1d0mXEOeqFKqhNQHz2
[INFO] Task total time: 17.097 s
[INFO] ------------------------------------------------------------------------
--
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
--
[INFO] Total time:  25.757 s
[INFO] Finished at: 2021-11-26T20:56:49+01:00
[INFO] ------------------------------------------------------------------------
--
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Generación war y subida - Artifactory)
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] withEnv
[Pipeline] {
[Pipeline] sh
+ mvn --settings settings.xml deploy -Dmaven.test.skip
[INFO] Scanning for projects...
[INFO]
[INFO] ---------------------------< tfm:directorio >-------------------------
--
[INFO] Building directorio Maven Webapp 1.0.0
[INFO] --------------------------------[ war ]---------------------------------
--
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:resources (default-resources) @
directorio ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 2 resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:compile (default-compile) @ directorio
---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:testResources (default-testResources)
@ directorio ---
[INFO] Not copying test resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:testCompile (default-testCompile) @
directorio ---
[INFO] Not compiling test sources
[INFO]
[INFO] --- maven-surefire-plugin:2.22.2:test (default-test) @ directorio ---
```

```
[INFO] Tests are skipped.
[INFO]
[INFO] --- maven-war-plugin:3.2.2:war (default-war) @ directorio ---
[INFO] Packaging webapp
[INFO] Assembling webapp [directorio] in
[/var/lib/jenkins/workspace/directorio-pipeline/target/directorio]
[INFO] Processing war project
[INFO] Copying webapp resources [/var/lib/jenkins/workspace/directorio-
pipeline/src/main/webapp]
[INFO] Webapp assembled in [435 msecs]
[INFO] Building war: /var/lib/jenkins/workspace/directorio-
pipeline/target/directorio.war
[INFO]
[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ directorio
---
[INFO] Installing /var/lib/jenkins/workspace/directorio-
pipeline/target/directorio.war to
/var/lib/jenkins/.m2/repository/tfm/directorio/1.0.0/directorio-1.0.0.war
[INFO] Installing /var/lib/jenkins/workspace/directorio-pipeline/pom.xml to
/var/lib/jenkins/.m2/repository/tfm/directorio/1.0.0/directorio-1.0.0.pom
[INFO]
[INFO] --- maven-deploy-plugin:2.8.2:deploy (default-deploy) @ directorio ---
Uploading to releases: https://tfm.jfrog.io/artifactory/default-maven-
local/tfm/directorio/1.0.0/directorio-1.0.0.war
Progress (1): 0.3/27 MB
Progress (1): 0.5/27 MB
Progress (1): 0.8/27 MB
Progress (1): 1.1/27 MB
Progress (1): 1.4/27 MB
Progress (1): 1.6/27 MB
Progress (1): 1.9/27 MB
Progress (1): 2.2/27 MB
Progress (1): 2.4/27 MB
Progress (1): 2.7/27 MB
Progress (1): 3.0/27 MB
Progress (1): 3.2/27 MB
Progress (1): 3.5/27 MB
Progress (1): 3.8/27 MB
Progress (1): 4.1/27 MB
Progress (1): 4.3/27 MB
Progress (1): 4.6/27 MB
Progress (1): 4.9/27 MB
Progress (1): 5.1/27 MB
Progress (1): 5.4/27 MB
Progress (1): 5.7/27 MB
Progress (1): 5.9/27 MB
Progress (1): 6.2/27 MB
Progress (1): 6.5/27 MB
Progress (1): 6.8/27 MB
Progress (1): 7.0/27 MB
Progress (1): 7.3/27 MB
Progress (1): 7.6/27 MB
Progress (1): 7.8/27 MB
Progress (1): 8.1/27 MB
Progress (1): 8.4/27 MB
Progress (1): 8.7/27 MB
Progress (1): 8.9/27 MB
Progress (1): 9.2/27 MB
Progress (1): 9.5/27 MB
Progress (1): 9.7/27 MB
Progress (1): 10/27 MB
Progress (1): 10/27 MB
```

```
Progress (1): 11/27 MB
Progress (1): 11/27 MB
Progress (1): 11/27 MB
Progress (1): 11/27 MB
Progress (1): 12/27 MB
Progress (1): 12/27 MB
Progress (1): 12/27 MB
Progress (1): 12/27 MB
Progress (1): 13/27 MB
Progress (1): 13/27 MB
Progress (1): 13/27 MB
Progress (1): 14/27 MB
Progress (1): 14/27 MB
Progress (1): 14/27 MB
Progress (1): 14/27 MB
Progress (1): 15/27 MB
Progress (1): 15/27 MB
Progress (1): 15/27 MB
Progress (1): 15/27 MB
Progress (1): 16/27 MB
Progress (1): 16/27 MB
Progress (1): 16/27 MB
Progress (1): 16/27 MB
Progress (1): 17/27 MB
Progress (1): 17/27 MB
Progress (1): 17/27 MB
Progress (1): 18/27 MB
Progress (1): 18/27 MB
Progress (1): 18/27 MB
Progress (1): 18/27 MB
Progress (1): 19/27 MB
Progress (1): 19/27 MB
Progress (1): 19/27 MB
Progress (1): 19/27 MB
Progress (1): 20/27 MB
Progress (1): 20/27 MB
Progress (1): 20/27 MB
Progress (1): 21/27 MB
Progress (1): 21/27 MB
Progress (1): 21/27 MB
Progress (1): 21/27 MB
Progress (1): 22/27 MB
Progress (1): 22/27 MB
Progress (1): 22/27 MB
Progress (1): 22/27 MB
Progress (1): 23/27 MB
Progress (1): 23/27 MB
Progress (1): 23/27 MB
Progress (1): 24/27 MB
Progress (1): 24/27 MB
Progress (1): 24/27 MB
Progress (1): 24/27 MB
Progress (1): 25/27 MB
Progress (1): 25/27 MB
Progress (1): 25/27 MB
Progress (1): 25/27 MB
Progress (1): 26/27 MB
Progress (1): 26/27 MB
Progress (1): 26/27 MB
Progress (1): 26/27 MB
Progress (1): 27/27 MB
Progress (1): 27/27 MB
```

```
Progress (1): 27 MB

Uploaded to releases: https://tfm.jfrog.io/artifactory/default-maven-
local/tfm/directorio/1.0.0/directorio-1.0.0.war (27 MB at 4.0 MB/s)
Uploading to releases: https://tfm.jfrog.io/artifactory/default-maven-
local/tfm/directorio/1.0.0/directorio-1.0.0.pom
Progress (1): 4.1/6.1 kB
Progress (1): 6.1 kB

Uploaded to releases: https://tfm.jfrog.io/artifactory/default-maven-
local/tfm/directorio/1.0.0/directorio-1.0.0.pom (6.1 kB at 13 kB/s)
Downloading from releases: https://tfm.jfrog.io/artifactory/default-maven-
local/tfm/directorio/maven-metadata.xml
Progress (1): 368 B

Downloaded from releases: https://tfm.jfrog.io/artifactory/default-maven-
local/tfm/directorio/maven-metadata.xml (368 B at 2.7 kB/s)
Uploading to releases: https://tfm.jfrog.io/artifactory/default-maven-
local/tfm/directorio/maven-metadata.xml
Progress (1): 320 B

Uploaded to releases: https://tfm.jfrog.io/artifactory/default-maven-
local/tfm/directorio/maven-metadata.xml (320 B at 792 B/s)
[INFO] ------------------------------------------------------------------------
--
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
--
[INFO] Total time:  20.114 s
[INFO] Finished at: 2021-11-26T20:57:17+01:00
[INFO] ------------------------------------------------------------------------
--
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Aprobación para despliegue)
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] withEnv
[Pipeline] {
[Pipeline] input
¿Se aprueba el despliegue?
Proceed or Abort
Approved by Lourdes
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Despliegue con Ansible)
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] withEnv
[Pipeline] {
[Pipeline] ansiColor
[Pipeline] {
```

```
[Pipeline] ansiblePlaybook
[directorio-pipeline] $ ansible-playbook playbook-directorio.yml -i
/etc/ansible/hosts --private-key /var/lib/jenkins/workspace/directorio-
pipeline/ssh17212611507835644471.key -u jenkins

PLAY [servidor]
*****************************************************************

TASK [Gathering Facts]
*********************************************************
[WARNING]: Platform linux on host 34.125.204.166 is using the discovered
Python
interpreter at /usr/bin/python3.7, but future installation of another Python
interpreter could change this. See
https://docs.ansible.com/ansible/2.9/referen
ce_appendices/interpreter_discovery.html for more information.
ok: [34.125.204.166]

TASK [Levantar tomcat si no lo esta]
*****************************************
ok: [34.125.204.166]

TASK [Desplegar directorio en tomcat]
*****************************************
changed: [34.125.204.166]

PLAY RECAP
*********************************************************************
34.125.204.166             : ok=3    changed=1    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0

[Pipeline] }

[Pipeline] // ansiColor
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

8.6 Arquitectura

En el siguiente diagrama se describen las máquinas y tecnologías utilizadas:

- Maquina local donde se encuentra:
    - Entorno de desarrollo basado en Eclipse, MySQL WorkBench y Git.
- Google Cloud Platform en donde se han creado:
    - Máquina SQL con MySQL para almacenar la base de datos.
    - Máquina Debian para desplegar la aplicación en un servidor de aplicaciones Tomcat.
- Instancia de Artifactory para almacenar los WAR generados.
- Proyecto en GitHub para versionar el código fuente.

Además, se describen con flechas los pasos llevados a cabo para desplegar la aplicación “Directorio” partiendo de un Pipeline en Jenkins:

1. Descarga del código de GitHub.
2. Compilación y ejecución de pruebas unitarias con Maven.
3. Análisis de código estático con SonarQube.
4. Subida de WAR a Artifactory.
5. Ejecución de Playbook de Ansible
6. Despliegue de aplicación en Tomcat.
7. Comunicación entre aplicación y base de datos.

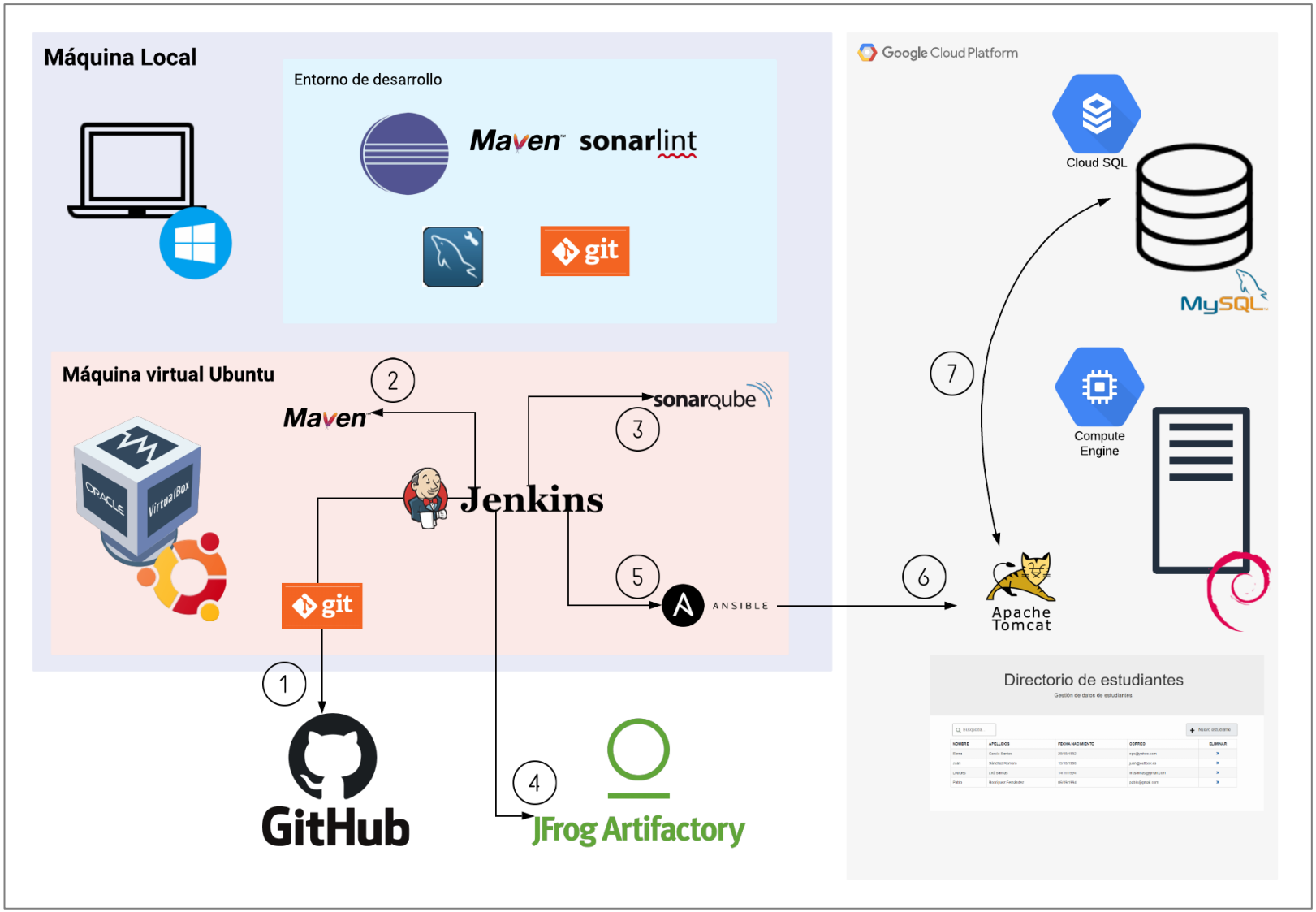

Imagen 63. Arquitectura, tecnologías y funcionamiento.

REFERENCIAS

[1] M. Fowler, «martinFowler.com,» 01 Mayo 2006. [En línea]. Available: https://martinfowler.com/articles/continuousIntegration.html. [Último acceso: 14 09 2021].

[2] S. M. a. A. G. Paul Duvall, Continuous Integration. Improving Software Quality and Reducing Risk., Addison-Wesley, 2007.

[3] L. Chen, «Continuous Delivery: Huge Benefits, but Challenges Too,» *IEEE Software. ,* vol. 32, nº 2, pp. 50 - 54, 2015.

[4] I. Sacolick, «InfoWorld,» 17 01 2020. [En línea]. Available: https://www.infoworld.com/article/3271126/what-is-cicd-continuous-integration-and-continuous-delivery-explained.html. [Último acceso: 27 11 2021].

[5] P. Riti, Pro DevOps with Google Cloud Platform, with Docker, Jenkins, and Kubernetes., Mullingar, Westmeath, Ireland: Apress, 2018.

[6] P. Debois, «Agile infrastructure and operations: how infra-gile are you?,» de *Agile Conference*, Toronto, Canada, 2008.

[7] M. B. A. v. B. A. C. W. C. M. F. J. G. J. H. A. H. R. J. J. K. B. M. R. C. M. S. M. K. S. J. S. y. D. T. Kent Beck, «https://agilemanifesto.org/,» 2001. [En línea]. Available: https://agilemanifesto.org/. [Último acceso: 18 09 2021].

[8] J. Highsmith, «History: The Agile Manifesto,» 2001 . [En línea]. Available: https://agilemanifesto.org/history.html. [Último acceso: 18 09 2021].

[9] Amazon Web Services, Inc. o sus empresas afiliadas., «aws,» 2021. [En línea]. Available: https://aws.amazon.com/es/devops/source-control/. [Último acceso: 19 09 2021].

[10] Free Software Foundation, «CVS—Concurrent Versions System v1.11.23,» 28 12 2015. [En línea]. Available: https://www.gnu.org/software/trans-coord/manual/cvs/cvs.html#What-is-CVS_003f. [Último acceso: 19 09 2021].

[11] B. S. Scott Chacon, «Getting Started - About Version Control,» de *Pro Git*, Apress, 2014, p. Chapter 1.1.

[12] The Apache Software Foundation, «Subversion,» 2018 . [En línea]. Available: https://subversion.apache.org/. [Último acceso: 19 09 2021].

[13] Git community, «git,» [En línea]. Available: https://git-scm.com/. [Último acceso: 19 09 2021].

[14] Mercurial community, [En línea]. Available: https://www.mercurial-scm.org/. [Último acceso: 19 09 2021].

[15] Perforce Software, Inc., «Perforce,» 2021. [En línea]. Available: https://www.perforce.com/solutions/version-control. [Último acceso: 02 10 2021].

[16] RhodeCode Inc., «RhodeCode,» 2016. [En línea]. Available: https://rhodecode.com/insights/version-control-systems-2016. [Último acceso: 02 10 2021].

[17] wikipedia, «Comparison of version-control software,» 02 09 2021. [En línea]. Available: https://en.wikipedia.org/wiki/Comparison_of_version-control_software. [Último acceso: 25 09 2021].

[18] Free Software Foundation, Inc., «GCC, the GNU Compiler Collection,» 16 09 2021. [En línea]. Available: https://gcc.gnu.org/. [Último acceso: 19 09 2021].

[19] Free Software Foundation, Inc., «GNU Operating System,» 19 01 2020. [En línea]. Available: https://www.gnu.org/software/make/. [Último acceso: 19 09 2021].

[20] The Apache Software Foundation, «The Apache Ant Project,» 13 07 2021. [En línea]. Available: https://ant.apache.org/. [Último acceso: 19 09 2021].

[21] The Apache Software Foundation, «Apache Maven Project,» 2021-2021. [En línea]. Available: https://maven.apache.org/. [Último acceso: 19 09 2021].

[22] Gradle Inc., «Gradle Build Tool,» 2021. [En línea]. Available: https://gradle.org/. [Último acceso: 19 09 2021].

[23] ISTQB, « ISTQB, International Software Testing Qualifications Board.,» 2018. [En línea]. Available: https://www.istqb.org/downloads/send/2-foundation-level-documents/281-istqb-ctfl-syllabus-2018-v3-1.html. [Último acceso: 19 09 2021].

[24] P. Hamill, Unit Test Frameworks: Tools for High-Quality Software Development, Sebastopol, California: O'Reilly Media, Inc., 2004.

[25] .NET Foundation, «xUnit.net,» 2021. [En línea]. Available: https://xunit.net/. [Último acceso: 19 09 2021].

[26] .NET Foundation, «nunit,» 2019. [En línea]. Available: https://nunit.org/. [Último acceso: 19 09 2021].

[27] Microsoft, «.NET,» 2021. [En línea]. Available: https://dotnet.microsoft.com/. [Último acceso: 19 09 2021].

[28] The JUnit Team, «JUnit,» 2021. [En línea]. Available: https://junit.org/junit5/. [Último acceso: 19 09 2021].

[29] Oracle and/or its affiliates, «Oracle,» 1993-2021. [En línea]. Available: https://docs.oracle.com/javase/8/docs/technotes/guides/language/index.html. [Último acceso: 19 09 2021].

[30] Eric Sommerlade, Michael Feathers, Jerome Lacoste , J.E. Hoffmann, Baptiste Lepilleur, Bastiaan Bakker, Steve Robb, «CppUnit Documentaion,» 2013. [En línea]. Available: http://cppunit.sourceforge.net/doc/1.8.0/. [Último acceso: 19 09 2021].

[31] cplusplus.com, «cplusplus,» 2000-2021. [En línea]. Available: https://www.cplusplus.com/. [Último

acceso: 19 09 2021].

[32] Python Software Foundation, «PyUnit,» 20 06 2019. [En línea]. Available: https://wiki.python.org/moin/PyUnit. [Último acceso: 19 09 2021].

[33] Python Software Foundation, «Python,» 2001-2021. [En línea]. Available: https://www.python.org/. [Último acceso: 19 09 2021].

[34] J. M. Tim Bacon, «XMLUnit,» 2020. [En línea]. Available: https://www.xmlunit.org/. [Último acceso: 19 09 2021].

[35] Textuality Services, Inc., «XML.com,» 1998-2008. [En línea]. Available: https://www.xml.com/. [Último acceso: 19 09 2021].

[36] bugScout, «bugscout,» 2020. [En línea]. Available: https://bugscout.io/es/. [Último acceso: 25 09 2021].

[37] SonarSource S.A, «SonarQube,» 2008-2021. [En línea]. Available: https://www.sonarqube.org/. [Último acceso: 25 09 2021].

[38] Kiuwan, «Kiuwan,» 2021. [En línea]. Available: https://www.kiuwan.com/. [Último acceso: 25 09 2021].

[39] Atlassian, «Bamboo,» 2021. [En línea]. Available: https://www.atlassian.com/software/bamboo. [Último acceso: 25 09 2021].

[40] Jenkins, «Jenkins,» [En línea]. Available: https://www.jenkins.io/. [Último acceso: 25 09 2021].

[41] Circle Internet Services, Inc., «circleci,» 2021. [En línea]. Available: https://circleci.com/. [Último acceso: 25 09 2021].

[42] JetBrains s.r.o., «TeamCity,» 2000-2021. [En línea]. Available: https://www.jetbrains.com/teamcity/. [Último acceso: 02 10 2021].

[43] Travis CI, «Travis CI,» [En línea]. Available: https://www.travis-ci.com/. [Último acceso: 02 10 2021].

[44] altexsoft, «altexsoft,» 19 02 2019. [En línea]. Available: https://www.altexsoft.com/blog/engineering/comparison-of-most-popular-continuous-integration-tools-jenkins-teamcity-bamboo-travis-ci-and-more/. [Último acceso: 02 10 2021].

[45] Red Hat, Inc., 2021. [En línea]. Available: https://www.ansible.com/. [Último acceso: 25 09 2021].

[46] M. Soni, DevOps for Web Development, Packt Publishing Ltd, 2016.

[47] B. Byrd, «DBmaestro,» 17 04 2019. [En línea]. Available: https://www3.dbmaestro.com/blog/7-vital-tools-for-devops-success. [Último acceso: 09 10 2021].

[48] M. Bravo, «opensource.com,» 12 12 2018. [En línea]. Available: https://opensource.com/article/18/12/configuration-management-tools. [Último acceso: 09 10 2021].

[49] Puppet, «Puppet,» 2021. [En línea]. Available: https://puppet.com/use-cases/continuous-configuration-automation/. [Último acceso: 03 10 2021].

[50] Progress Software Corporation and/or its subsidiaries or affiliates., «Chef,» 2021. [En línea]. Available: https://www.chef.io/products/chef-infrastructure-management. [Último acceso: 03 10 2021].

[51] Ansible project contributors, «Docs Ansible,» 21 12 2021. [En línea]. Available: https://docs.ansible.com/ansible/latest/user_guide/playbooks_intro.html. [Último acceso: 09 01 2022].

[52] Northern.tech AS, «CFEngine,» 2021. [En línea]. Available: https://cfengine.com/product-overview/. [Último acceso: 03 10 2021].

[53] Red Hat, Inc., «Integraciones de Ansible,» 2020. [En línea]. Available: https://www.ansible.com/integrations. [Último acceso: 09 10 2021].

[54] Wikipedia, «wikipedia,» 2021. [En línea]. Available: https://en.wikipedia.org/wiki/Comparison_of_open-source_configuration_management_software. [Último acceso: 09 10 2021].

[55] Flexera™, «RightScale 2019 State of the Cloud Report from Flexera,» 2019. [En línea]. Available: https://resources.flexera.com/web/media/documents/rightscale-2019-state-of-the-cloud-report-from-flexera.pdf. [Último acceso: 09 10 2021].

[56] A. Safonov, «MEREHEAD,» 25 11 2021. [En línea]. Available: https://merehead.com/blog/chef-vs-puppet-vs-ansible-vs-saltstack-better-2022/. [Último acceso: 27 11 2021].

[57] E. DeBoer, «sonatype,» 30 01 2020. [En línea]. Available: https://blog.sonatype.com/why-do-i-need-a-binary-repository-manager. [Último acceso: 26 09 2021].

[58] JFrog Ltd, «JFrog Artifactory,» 2021. [En línea]. Available: https://jfrog.com/artifactory/. [Último acceso: 25 09 2021].

[59] Sonatype, Inc., «nexus repository pro,» 2008-2021. [En línea]. Available: https://www.sonatype.com/products/repository-pro. [Último acceso: 25 09 2021].

[60] The Apache Software Foundation., «Archiva,» 19 06 2020. [En línea]. Available: https://archiva.apache.org/. [Último acceso: 25 09 2021].

[61] sonatype, «Sonatype,» [En línea]. Available: https://help.sonatype.com/repomanager3/formats. [Último acceso: 02 10 2021].

[62] D. Bereznitsky, «JFrog Artifactory Vs. Sonatype Nexus – The Integration Matrix,» 5 02 2021. [En línea]. Available: https://jfrog.com/blog/artifactory-vs-nexus-integration-matrix/. [Último acceso: 27 11 2021].

[63] CD Foundation The Linux Foundation, «CD Foundation,» 2021. [En línea]. Available: https://cd.foundation/projects/. [Último acceso: 26 09 2021].

[64] Jenkins, «Plugins Jenkins,» 2022. [En línea]. Available: https://plugins.jenkins.io/. [Último acceso: 12 01 2022].

[65] Git, «About Git,» [En línea]. Available: https://git-scm.com/about/branching-and-merging. [Último acceso: 28 11 2021].

[66] The Apache Software Foundation, «Introduction to the Build Lifecycle,» [En línea]. Available: https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html. [Último acceso: 12 01

2022].

[67] The Apache Software Foundation, «Introduction to the Dependency Mechanism,» [En línea]. Available: https://maven.apache.org/guides/introduction/introduction-to-dependency-mechanism.html. [Último acceso: 12 01 2022].

[68] Eclipse Foundation, «Eclipse,» [En línea]. Available: https://www.eclipse.org/. [Último acceso: 03 10 2021].

[69] The Apache Software Foundation, «Apache Tomcat,» 1999-2021. [En línea]. Available: https://tomcat.apache.org/download-90.cgi. [Último acceso: 07 11 2021].

[70] Sun Microsystems, Inc. , «Sun ONE,» 2003. [En línea]. Available: https://docs.oracle.com/cd/E19199-01/816-6774-10/a_war.html. [Último acceso: 09 01 2022].

[71] Google, «Google Cloud,» [En línea]. Available: https://cloud.google.com/?hl=es. [Último acceso: 27 11 2021].

[72] SPI, «Debian,» 1997-2021. [En línea]. Available: https://www.debian.org/index.es.html. [Último acceso: 28 11 2021].

[73] Google, «Google Cloud,» 05 11 2021. [En línea]. Available: https://cloud.google.com/compute/docs/images/os-details?hl=es. [Último acceso: 27 11 2021].

[74] Google, «Google Cloud,» [En línea]. Available: https://cloud.google.com/sql-server/?hl=es. [Último acceso: 27 11 2021].

[75] Jenkins, «Creating your first Pipeline,» [En línea]. Available: https://www.jenkins.io/doc/pipeline/tour/hello-world/#what-is-a-jenkins-pipeline. [Último acceso: 09 01 2022].

[76] M. Fowler, «Domain Specific Languages,» 2010. [En línea]. Available: https://martinfowler.com/books/dsl.html. [Último acceso: 09 01 2022].

[77] Jenkins, «Pipeline Syntax,» [En línea]. Available: https://www.jenkins.io/doc/book/pipeline/syntax/. [Último acceso: 28 11 2021].

[78] Jenkins, «Using a Jenkinsfile,» [En línea]. Available: https://www.jenkins.io/doc/book/pipeline/jenkinsfile/. [Último acceso: 28 11 2021].

[79] Jenkins, «Installing Jenkins. Linux.,» [En línea]. Available: https://www.jenkins.io/doc/book/installing/linux/. [Último acceso: 10 11 2021].

[80] Jenkins, «Using local language,» [En línea]. Available: https://www.jenkins.io/doc/book/using/using-local-language/. [Último acceso: 10 11 2021].

[81] GitHub, Inc., «GitHub,» 2021. [En línea]. Available: https://github.com. [Último acceso: 07 11 2021].

[82] Atlassian, «Git Bash,» [En línea]. Available: https://www.atlassian.com/git/tutorials/git-bash. [Último acceso: 07 11 2021].

[83] Eclipse Foundation, Inc., «EGit/User Guide,» [En línea]. Available:

https://wiki.eclipse.org/EGit/User_Guide. [Último acceso: 07 11 2021].

[84] GitHub, Inc., «GitHub Docs,» 2021. [En línea]. Available: https://docs.github.com/en/authentication/connecting-to-github-with-ssh. [Último acceso: 07 11 2021].

[85] Git , «Getting Started - First-Time Git Setup,» 2021. [En línea]. Available: https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup. [Último acceso: 07 11 2021].

[86] Git, «Git,» [En línea]. Available: https://git-scm.com/book/es/v2/Ramificaciones-en-Git-%C2%BFQu%C3%A9-es-una-rama%3F. [Último acceso: 09 01 2022].

[87] Git, «Git,» [En línea]. Available: https://git-scm.com/docs/git-merge. [Último acceso: 09 01 2022].

[88] Oracle Corporation and/or its affiliates, «MySQL Community Downloads,» 2021. [En línea]. Available: https://dev.mysql.com/downloads/mysql/5.7.html. [Último acceso: 07 11 2021].

[89] Oracle Corporation and/or its affiliates, «MySQL,» 2021. [En línea]. Available: https://www.mysql.com/products/workbench/. [Último acceso: 07 11 2021].

[90] Eclipse Foundation, «Eclipse IDE for Java Developers,» 18 03 2020. [En línea]. Available: https://www.eclipse.org/downloads/packages/release/2020-03/r/eclipse-ide-java-developers. [Último acceso: 07 11 2021].

[91] Bootstrap team, «getboostrap,» [En línea]. Available: https://getbootstrap.com/. [Último acceso: 07 11 2021].

[92] RedHat, «Hibernate,» 2021. [En línea]. Available: https://hibernate.org/. [Último acceso: 07 11 2021].

[93] VMware, Inc. or its affiliates, «spring,» 2021. [En línea]. Available: https://spring.io/. [Último acceso: 07 11 2021].

[94] Google, «Ventajas de Google Cloud,» [En línea]. Available: https://cloud.google.com/why-google-cloud?hl=es. [Último acceso: 28 11 2021].

[95] The Apache Software Foundation, «Installing Apache Maven,» 2002-2021. [En línea]. Available: https://maven.apache.org/install.html. [Último acceso: 10 11 2021].

[96] S. F. a. friends., «mockito,» [En línea]. Available: https://site.mockito.org/. [Último acceso: 07 11 2021].

[97] Junta de Andalucía, «MADEJA - Plantilla Plan de Pruebas Funcionales,» 14 04 2011. [En línea]. Available: https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/462. [Último acceso: 07 11 2021].

[98] SonarSource S.A., «SonarQube Docs 9.2. Install Server.,» 2021. [En línea]. Available: https://docs.sonarqube.org/latest/setup/install-server/. [Último acceso: 23 11 2021].

[99] T. Rakwach, «VULTR,» 16 06 2021. [En línea]. Available: https://www.vultr.com/docs/install-sonarqube-on-ubuntu-20-04-lts. [Último acceso: 23 11 2021].

[100] SonarSource S.A, «SonarQube Docs 9.2. Requirements.,» 2021. [En línea]. Available: https://docs.sonarqube.org/latest/requirements/requirements/. [Último acceso: 23 11 2021].

[101] SonarSource S.A., «sonarlint,» 2008-2021. [En línea]. Available: https://www.sonarlint.org/eclipse. [Último acceso: 23 11 2021].

[102] JFrog Ltd, «Jfrog,» 2021. [En línea]. Available: https://jfrog.com/start-free/#saas. [Último acceso: 21 11 2021].

[103] A. p. contributors, «Ansible Documentation. Installing Ansible.,» 12 11 2021. [En línea]. Available: https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#installing-ansible-on-ubuntu. [Último acceso: 23 11 2021].

[104] Github, «Generating a new SSH key and adding it to the ssh-agent,» [En línea]. Available: https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent. [Último acceso: 10 11 2021].

[105] GitHub, Inc., «GitHub Docs,» 2021. [En línea]. Available: https://docs.github.com/en/developers/webhooks-and-events/webhooks/about-webhooks. [Último acceso: 29 11 2021].

[106] K. M. y. O. N. Kanstantsin Shautsou, «Plugins Jenkins: GitHub.,» 01 09 2021. [En línea]. Available: https://plugins.jenkins.io/github/. [Último acceso: 29 11 2021].

[107] A. Atzmony, «Jfrog Confluence. Maven Repository.,» 04 2021. [En línea]. Available: https://www.jfrog.com/confluence/display/JFROG/Maven+Repository#MavenRepository-DeployingArtifactsThroughArtifactory. [Último acceso: 22 11 2021].

[108] E. B. Salomon, «Permissions,» 24 06 2021. [En línea]. Available: https://www.jfrog.com/confluence/display/JFROG/Permissions. [Último acceso: 22 11 2021].

[109] E. E. Jean-Christophe Sirot, «Plugins Jenkins. Ansible.,» 30 10 2020. [En línea]. Available: https://plugins.jenkins.io/ansible/. [Último acceso: 23 11 2021].

[110] Jenkins, «Defining a pipeline in scm,» [En línea]. Available: https://www.jenkins.io/doc/book/pipeline/getting-started/#defining-a-pipeline-in-scm. [Último acceso: 10 11 2021].

[111] L. Newman, «Declarative Pipeline for Maven Projects,» 07 02 2017. [En línea]. Available: https://www.jenkins.io/blog/2017/02/07/declarative-maven-project/. [Último acceso: 10 11 2021].

[112] SonarSource S.A, «Analyzing. SonarQube Docs.,» 30 04 2021. [En línea]. Available: https://docs.sonarqube.org/latest/analysis/scan/sonarscanner-for-maven/. [Último acceso: 28 11 2021].

[113] Julien Oster, Daniel Doubrovkine, Jonathan Süssemilch Poulain, Joe Merten, Mike Kobit, Devin Nusbaum, Tomasz Szmytka, 08 11 2021. [En línea]. Available: https://plugins.jenkins.io/ansicolor/. [Último acceso: 28 11 2021].

[114] Jenkins, «Pipeline: Basic Steps,» [En línea]. Available: https://www.jenkins.io/doc/pipeline/steps/workflow-basic-steps/#catcherror-catch-error-and-set-build-result-to-failure. [Último acceso: 29 11 2021].

[115] C. Bradley, «Scrum,» 09 07 2015. [En línea]. Available: https://www.scrum.org/resources/blog/scrum-who-are-key-stakeholders-should-be-attending-every-sprint-review. [Último acceso: 29 11 2021].

[116] Selenium, «Selenium,» 2021. [En línea]. Available: https://www.selenium.dev/. [Último acceso: 25 09 2021].

[117] KAtalon, Inc., «Katalon,» 2021. [En línea]. Available: https://www.katalon.com/. [Último acceso: 25 09 2021].

[118] Robot Framework, «Robot Framework,» [En línea]. Available: https://robotframework.org/. [Último acceso: 25 09 2021].

[119] Cypress.io, «cypress,» 2021. [En línea]. Available: https://www.cypress.io/. [Último acceso: 02 10 2021].

[120] Apache Software Foundation, «Apache JMeter,» 2021. [En línea]. Available: https://jmeter.apache.org/. [Último acceso: 29 11 2021].

[121] Liquibase Inc., 2021. [En línea]. Available: https://www.liquibase.org/. [Último acceso: 25 09 2021].

[122] DBmaestro, «DBmaestro. DevOps for Database.,» [En línea]. Available: https://www.dbmaestro.com/database-source-control. [Último acceso: 25 09 2021].

[123] PyInstaller Development Team, «PyInstaller,» 11 08 2021. [En línea]. Available: https://www.pyinstaller.org/. [Último acceso: 19 09 2021].

[124] F. Toledo, Introducción a las Pruebas de Sistemas de Información., Abstracta, Montevideo, Uruguay: Abstracta, 2014.

[125] S. Bergmann, «PHPUnit,» [En línea]. Available: https://phpunit.de/. [Último acceso: 19 09 2021].

[126] The PHP Group, «php,» 2001-2021. [En línea]. Available: https://www.php.net/. [Último acceso: 19 09 2021].

[127] Typemock Inc., «Typemock,» 2006. [En línea]. Available: https://www.typemock.com/. [Último acceso: 19 09 2021].

[128] Microsoft, «Microsoft,» 2021. [En línea]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/. [Último acceso: 19 09 2021].

[129] M. V. Webcraft, LLC, «versionSQL,» 2021. [En línea]. Available: https://www.versionsql.com/. [Último acceso: 25 09 2021].

[130] Red Gate Software Ltd, «Source Control for Oracle,» 1999-2021. [En línea]. Available: https://www.red-gate.com/products/oracle-development/source-control-for-oracle/. [Último acceso: 25 09 2021].

[131] Microsoft, «SQL Server Management Studio (SSMS),» 16 09 2021. [En línea]. Available: https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15. [Último acceso: 25 09 2021].

[132] npm, Inc, «npm,» [En línea]. Available: https://www.npmjs.com/. [Último acceso: 25 09 2021].

[133] Docker, Inc., «Docker,» [En línea]. Available: https://www.docker.com/. [Último acceso: 26 09 2021].

[134] Amazon, «aws,» [En línea]. Available: https://aws.amazon.com/. [Último acceso: 26 09 2021].

[135] Microsoft, «Azure,» 2021. [En línea]. Available: https://azure.microsoft.com/. [Último acceso: 26 09 2021].

[136] Google, «Google Cloud,» [En línea]. Available: https://cloud.google.com/. [Último acceso: 26 09 2021].

[137] The Kubernetes Authors, «kubernetes,» 2021. [En línea]. Available: https://kubernetes.io/. [Último acceso: 26 09 2021].

[138] E. Noble, «Deployment,» de *Pro T-SQL 2019*, Berkeley, CA, Apress, 2020, p. 299.